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

Solución óptima de modelos dinámicos detallados de robots industriales basados en la norma L1

Máster Universitario en
Ingeniería de Materiales y Fabricación

Trabajo Fin de Máster

Autor: José David Fuentes Lárez

Directores: Javier Ros y Javier Faulin.

Pamplona, 11 de Septiembre de 2018

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen:

El uso de la dinámica de sistemas multicuerpo, permite la simulación de sistemas mecánicos, en ella se logran describir las fuerzas y desplazamientos aplicados, el modelo, queda expresado en términos de los parámetros del sistema (tales como masas, inercias, geometrías, etc…).

El resolver estos sistemas de manera exacta, puede conllevar una carga computacional alta, haciendo que trabajar en tiempo real, para aplicaciones de control, o de filtrado sea complicado. La posibilidad de resolver estos sistemas de manera acelerada, resulta un tema de interés dentro del mundo de la dinámica de sistemas multicuerpo, en general se requiere simplificar o reducir el modelo. Sin embargo, estas técnicas disminuyen la precisión de los modelos.

Una manera de reducir estos modelos, es ir quitando aquellos parámetros cuya contribución al modelo sea pequeña (que el error cometido al quitar el parámetro sea el más pequeño), sin embargo los parámetros no son independientes entre sí y por tanto el problema se vuelve un problema combinatorio.

En este trabajo, se proponen dos metodologías de reducción denominadas Norma L1 y Norma L1 iterativo, siendo comparadas con otras actualmente desarrolladas. El algoritmo no-lineal usado para la solución de problemas convexos bajo la norma L1 es el desarrollado en [1].

Este trabajo es una contribución a la investigación hecha en [2] sobre “simplification of multibody models by parameter reduction”

Palabras claves: Norma L1, Dinámica de Sistema Multicuerpos, Reducción de modelos, selección de modelos, parámetros base, estimación de parámetros, optimizacion convexa.

Abstract:

The use of the Multibody Dynamics Systems (MDS), it is possible the simulation of dynamics systems, in it, you can describe the forces and displacements applied, the model is express by the system parameters (such as mass, inertia, geometry, etc…)

Solving the dynamics systems precisely, has a high computational burden, making that work in real time for filters or control applications complex. Being able to speed up, the way to solve these systems becomes interesting to the MDS, but, by doing so the model decrease their precision.

One way of reducing these dynamic systems, is taking out those parameters which are meaningless to the model, nevertheless these parameters are not independent so it becomes a combinatorial problem.

In this work, it is propose two methodologies to reduce these models called: L1 norm, iterative L1 norm, being compared with other methodologies actually develop. The non-lineal algorithm to solve convex problem by L1 norm is the one used in [1]. This job is a contribution to [2] about “simplification of multibody models by parameter reduction”

Keywords: L1 norm, Multibody Dynamics, Model Reduction, Model Selection, Base parameters, Parameters estimation, convex optimization.

Índice General

Índice de tablas:

Índice de Figuras

1. Introducción

La mecánica es la parte de la física que estudia el movimiento y el equilibrio de los cuerpos, así como las fuerzas que los producen. Un sistema mecánico se puede definir como un sistema constituido fundamentalmente por componentes, dispositivos o elementos que tienen como función específica transformar, o transmitir el movimiento desde las fuentes que lo generan, estos se caracterizan por presentar elementos, o piezas sólidas, con el objeto de realizar movimientos por acción o efecto de una fuerza.

Un sistema multicuerpo es la modelización de estos sistemas mecánicos, el estudio de la dinámica (rama de la física que describe la evolución en el tiempo de un sistema físico en relación con los motivos o causas que provocan los cambios de estado físico o estado de movimiento), de estos sistemas es a lo que se le denomina Dinámica de sistema Multicuerpos. Las técnicas de la Dinámica de sistemas multicuerpo (DSM) permiten la simulación de cualquier sistema o subsistema mecánico, y con ello su análisis, diseño y mejora.

En la actualidad, la simulación numérica de sistemas dinámicos es un tema de interés especial para el mundo científico puesto que es usada en la industria, específicamente en el desarrollo de robots industriales, (manipulador programable en tres o más ejes multipropósito, controlado automáticamente y reprogramable) para predecir los efectos de los esfuerzos y cargas.

El requerimiento de simulaciones que aceleren el diseño o que permitan ser utilizadas en aplicaciones en tiempo real ha sido una de las principales preocupaciones de los investigadores desde los años 90, en el área de la dinámica de sistemas multicuerpos. Por tanto, la reducción de modelos se ha vuelto un tema fundamental en la literatura de la dinámica de sistemas multicuerpos (DSM). En este contexto, la simplificación de modelos frecuentemente significa la reducción de orden, que se suele expresar como, una disminución en el número de grados de libertad del modelo intentando no comprometer su precisión.

El desempeño en tiempo real es un requerimiento en muchas aplicaciones de sistemas multicuerpos, como en la de alimentar el sistema de control predictivo o Feed-foward (Sistema pre-alimentado: son sistemas que reaccionan de acuerdo a variables que vienen de su entorno), [3], el control basado en modelos híbridos (modelos de control analíticos y basados en datos) [4], modelos basados en el filtrado no-lineal [5] [6] , modelos basados en la monitorización de su condición (condition monitoring) [7] ,etc. Frecuentemente, los modelos no requieren que la flexibilidad a nivel de cuerpos, sea tomada en cuenta y por lo tanto la reducción de orden del modelo deja de ser lo más apropiado. Sin embargo, debido a las demandas de desempeño en tiempo real, los investigadores mantienen los esfuerzos para minimizar la demanda

computacional requerida por los modelos. En este contexto, formulaciones que integran las ecuaciones dinámicas exactas usando un número mínimo de operaciones de tipo $O(n^3)$ y formulaciones de complejidad tipo $O(n)$ desempeñan un rol predominante [8].

El trabajo presentado a continuación parte de [2] en donde se proponen diferentes técnicas de selección de modelos para obtener modelos de sistemas multicuerpos de parámetros reducidos. El problema de regresión es alimentado con datos obtenidos de simulaciones basadas en el modelo exacto y las técnicas de selección de modelos son usadas para elegir el conjunto de parámetros más pequeños que se ajusten a esos datos para una tolerancia de error dada.

En este trabajo además de incorporar las propuestas de reducción de modelos establecidas en [2] [9], con fines meramente comparativos, también explora otros métodos alternativos tales como la eliminación de parámetros bajo la Norma L1 partiendo de la solución de problemas convexos expuestos en [1], la aplicación de la Norma L1 iterativa, LASSO y FS iterativo. Aunque en este último la idea es presentada en [2], sin embargo su desarrollo e implementación es descrita en esta memoria.

2. Justificación y objetivos:

2.1 Objetivos:

2.1.1 Objetivo principal:

El objeto de este proyecto es, mediante diferentes técnicas, hallar un método eficiente, en el cual se pueda reducir el tamaño de los modelos dinámicos mediante la eliminación parámetros, buscando obtener el menor error posible.

2.1.2 Objetivos específicos:

1. Buscar en la literatura la aplicación de técnicas para reducir la complejidad numérica de distintas aplicaciones.
2. La aplicación de estas técnicas de reducción a modelos dinámicos representativos.
3. Hallar el error que se tiene al aplicar cada una de las técnicas seleccionadas.
4. Comparación y análisis de cada una de las técnicas de reducción

Los aspectos más importantes en el desarrollo del proyecto son:

1. Análisis de distintas metodologías.
2. Selección de métodos.
3. Implementación de los distintos métodos en MATLAB.
4. Conclusiones.

En este proyecto se ha realizado en su gran mayoría mediante la creación de algoritmos para su uso en MATLAB, aunque a día de hoy existen diferentes programas, MATLAB contiene un conjunto de paquetes de optimización y de funciones desarrolladas para casi todo tipo de aplicaciones en este caso el algoritmo usado para resolver los problemas de optimización convexa expuesto en [1] ha sido desarrollado en MATLAB, también los paquetes de funciones provenientes del Lib3D_MEC_GiNaC para los cálculos relacionados con la DSM, por tanto en este trabajo se ha optado por el uso de este programa para su desarrollo.MATLAB se establece como el programa de uso estándar de facto en la ingeniería.

MATLAB contiene un entorno de escritorio ajustado para flujos de trabajos científicos y de ingeniería, también posee documentación escrita, soporte y foros para ingenieros. Las funcionalidades desarroladas en MATLAB son probadas rigurosamente y totalmente documentadas, MATLAB en el ámbito científico posee un alto grado de confiabilidad, es usado en diferentes ámbitos como en la industria aeroespacial, o medica. MATLAB tiene un equipo de ingenieros que constantemente verifica la calidad del software ejecutando millones de pruebas sobre el código base todos los días.

3. Estado del arte

3.1 Sistemas multicuerpos:

Las técnicas de la Dinámica de sistemas multicuerpo (DSM) permiten la simulación de cualquier sistema o subsistema mecánico, y con ello su análisis, diseño y mejora. Resulta claro en este ámbito el interés industrial, económico y científico de la DSM.

Estos modelos son obtenidos en función de los parámetros (tales como masas, inercias, geometrías, etc…), de los cuales se logran describir los movimientos y las fuerzas aplicadas en un determinado tiempo o instante. Por tanto, los modelos dinámicos exactos en el mundo de la robótica son requeridos para el control, o para simular las cargas presentes en los componentes.

3.2 Modelado de sistemas mecánicos:

El modelado de sistemas mecánicos es un campo que lleva años en la industria y que hoy en día se encuentra bastante consolidado. Al día de hoy existe un amplio número de libros en los que se describe tanto el modelado cinemático, como el modelado dinámico de sistemas mecánicos. [10], [11].

Cabe mencionar que para el modelo dinámico de un sistema robótico partimos del propuesto por [12], el cual viene dado por la siguiente ecuación:

$$K_{z\varphi}(z, \dot{z}, \ddot{z})\varphi = \tau_z \qquad 1$$

Donde $K_{z\varphi}$ representa la matriz de observación del sistema que contiene la información de la posición, velocidad y aceleración $(z, \dot{z}, \ddot{z})$, que corresponden con las coordenadas generales o acota el mecanismo y τ_z representa el vector de fuerzas y momentos externos que actúan sobre el sistema. El vector φ, incluye los parámetros dinámicos del sistema.

La matriz que se crea a partir de las ecuaciones dinámicas del movimiento en cada estado, viene denotada por $W(\mathcal{E})\varphi$ y se le conoce como matriz de observación para un conjunto de datos $\mathcal{E}$.

$$W(\mathcal{E})\varphi = \begin{bmatrix} K_{z\varphi}(z^i, \dot{z}^i, \ddot{z}^i) \\ \dots \\ K_{z\varphi}(z^n, \dot{z}^n, \ddot{z}^n) \end{bmatrix} \varphi = \begin{Bmatrix} {\tau_z}^i \\ \dots \\ {\tau_z}^n \end{Bmatrix} = X(\mathcal{E}) \qquad 2$$

3.3 Métodos de selección de modelos reducidos:

Los métodos de selección de modelos reducidos, son un conjunto de técnicas usadas en diferentes contextos científicos para representar un conjunto de datos en términos de un número reducido de parámetros, sacrificando en el proceso la precisión con la cual el conjunto de datos es representado.

Aparentemente, en el área de la dinámica de sistemas multicuerpos, estos métodos no han sido enteramente explorados.

Por otra parte los modelos multicuerpo pueden ser considerados paramétricos en términos de sus parámetros dinámicos por lo que es posible aplicar diferentes técnicas para reducir este número de parámetros.

Estos modelos de parámetros reducidos logran tener una complejidad computacional inferior al original. Sin embargo, pueden llegar a mantener un nivel de precisión deseado.

Los métodos de reducción de parámetros [12], [2] planteados en la dinámica de sistemas multicuerpos parten de un cierto número de observaciones hechas a un sistema mecánico, las cuales son expresadas de la siguiente manera:

$$d_z(z, \dot{z}, \ddot{z}, \varphi) = \tau_z \qquad 3$$

Donde, d_z es el llamado modelo dinámico inverso, con él, es posible obtener el vector de fuerzas que son aplicadas externamente (τ_z) a partir, de los estados del sistema $(z, \dot{z}, \ddot{z})$ y el vector de parámetros dinámicos φ. Un sistema dinámico puede ser expresado de la siguiente manera:

$$d_z(z, \dot{z}, \ddot{z}, \varphi) = M_{zz}(z, \varphi)\ddot{z} + \delta_z(z, \dot{z}, \varphi) = \tau_z \qquad 4$$

Donde M_{zz} representa la función de matriz de masa y δ_z representa las fuerzas centrífugas, constitutivas y de Coriolis.

Teniendo en cuenta la linealidad de d_z con respecto de los parámetros dinámicos esta ecuación se puede expresar:

$$d_z(z, \dot{z}, \ddot{z}, \varphi) = K_{z\varphi}(z, \dot{z}, \ddot{z})\varphi = \tau_z \quad 5$$

Esta es la manera estándar expresada para la estimación de parámetros dinámicos, donde $K_{z\varphi}$ es la matriz de observación. El objetivo de la simplificación por reducción de parámetros es definido en este contexto como la manera de eliminar algunos parámetros de φ y las correspondientes columnas de $K_{z\varphi}$ para que el vector τ_z pueda ser aproximado como una combinación lineal de las columnas.

$$K_{z\varphi_R}(z, \dot{z}, \ddot{z})\varphi_R \approx K_{z\varphi}(z, \dot{z}, \ddot{z})\varphi = \tau_z \quad 6$$

Por lo que φ_R se establece como el conjunto de parámetros reducidos y $K_{z\varphi_R}$ es la matriz de observación reducida. Para propósitos de implementación es conveniente eliminar los parámetros directamente de la expresión d_z lo que hace que:

$$d_z(z, \dot{z}, \ddot{z}, \varphi) = d_z(z, \dot{z}, \ddot{z}, \varphi_R) \quad 7$$

En el contexto de la identificación de sistemas robóticos, el modelo es frecuentemente re-parametrizado en términos de un número más pequeño de parámetros dinámicos.

Por lo tanto la estimación de parámetros es ajustada asegurando que las ecuaciones dinámicas se satisfacen para un conjunto de observaciones o de estimaciones, $\mathcal{E} = \{(z^i, \dot{z}^i, \ddot{z}^i, {\tau_z}^i) | i = 1, \ldots, n_\mathcal{E}\}$:

$$W(\mathcal{E})\varphi = \begin{bmatrix} K_{z\varphi}(z^i, \dot{z}^i, \ddot{z}^i) \\ \ldots \\ K_{z\varphi}(z^n, \dot{z}^n, \ddot{z}^n) \end{bmatrix} \varphi = \begin{Bmatrix} {\tau_z}^i \\ \ldots \\ {\tau_z}^n \end{Bmatrix} = X(\mathcal{E}) \quad 8$$

Donde $W(\mathcal{E})$ (como se ha establecido anteriormente) es la llamada matriz de observación para un conjunto de mediciones $\mathcal{E}$. En general, existen dependencias lineales entre las columnas $W(\mathcal{E})$, por lo tanto, el conjunto de parámetros φ no es independiente de $W(\mathcal{E})$ por lo que $W(\mathcal{E})$ es reordenado de la siguiente manera:

$$[W_{\varphi_R} \quad W_{\varphi_E}] \quad 9$$

Donde, la matriz W_{φ_R} está hecha de un conjunto de parámetros independientes y W_{φ_E} está formada por los parámetros dependientes. Por lo que, la matriz de parámetros es expresada como:

$$\varphi = \begin{bmatrix} \varphi_R \\ \varphi_E \end{bmatrix} \quad 10$$

Donde, φ_R es el conjunto de parámetros independientes asociados con W_{φ_R} y φ_E es el conjunto de parámetros dependientes o "excluidos" asociados a W_{φ_E}. Las columnas W_{φ_E} pueden ser expresadas como una combinación lineal de la matriz W_{φ_R} con la siguiente relación:

$$W_{\varphi_E} = W_{\varphi_R}\beta_{\varphi_R} \quad 11$$

En estas expresiones las columnas de la matriz β_{φ_R} contienen los coeficientes de dichas combinaciones lineales, por lo que el problema es reescrito de la siguiente forma:

$$[W_{\varphi_R} \quad W_{\varphi_E}] \begin{Bmatrix} \varphi_R \\ \varphi_E \end{Bmatrix} = [W_{\varphi_R} \quad W_{\varphi_R}\beta_{\varphi_R}] \begin{Bmatrix} \varphi_R \\ \varphi_E \end{Bmatrix} = W_{\varphi_R}(\varphi_R + \beta_{\varphi_R}\varphi_E) = W_{\varphi_R}{\varphi'}_R \quad 12$$

Lo que nos lleva al problema inicial para la identificación de parámetros:

$$W_{\varphi_R}(\mathcal{E}){\varphi'}_R = X(\mathcal{E}) \quad 13$$

Donde:

$${\varphi'}_R(\mathcal{E}) = \varphi_R + \beta_{\varphi_R}(\mathcal{E})\varphi_E \quad 14$$

Es el llamado conjunto de parámetros base o parámetros mínimos.

Por lo tanto el problema según [2] pasa a ser expresado como:

"Para un modelo dinámico dado, con un conjunto de parámetros conocidos, se busca determinar el mínimo conjunto de parámetros del modelo que logre aproximar los datos del modelo, con una deseada precisión".

Para la resolución de este problema se establecen dos preguntas por definir las cuales son:

1. ¿Cuáles son los conjuntos de parámetros característicos (de los que se define enteramente mi sistema y donde se considera exacto)?

2. ¿Cómo se establece la medida del error?

Por lo que debido a esto se establecieron las siguientes bases:

1. La estimación característica de un conjunto de datos es sistemática, dependiente de cada aplicación, y debería caracterizar el rango dinámico del sistema entero en la aplicación considerada. Este ajuste, $\varepsilon = \{(z^i, \dot{z}^i, \ddot{z}^i)|i = 1, \dots, n_\varepsilon\}$ puede ser obtenido mediante los datos de muestra del modelo entero.

Por ejemplo, en el caso de sistemas multicuerpos con todos sus grados de libertad actuados, las trayectorias son parametrizadas para un conjunto de coordenadas independientes, y un criterio de optimización es usado para determinar los valores de los parámetros. Una comparación de algunos de los criterios clásicos para la optimización de trayectorias puede ser encontrada en [13]. La parametrización de trayectorias basadas en series armónicas [14] [15] [16] y en polinómicas [17] [18] han sido propuestas en la literatura. Para sistemas sub-actuados, las simulaciones dinámicas tienen que ser ejecutadas para un conjunto de simulaciones representativas de trabajo. La validación del conjunto de datos $\mathrm{v} = \{(z^i, \dot{z}^i, \ddot{z}^i, \tau_z^i)|i = 1, \dots, n_\mathrm{v}\}$ puede ser obtenida usando el mismo procedimiento.

2. Para un conjunto dado de parámetros reducidos del modelo, el siguiente error de criterio o medición de predicción de error para la identificación de parámetros dinámicos es propuesto:

$$\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) = \frac{\left\|\sum^{-1/2}\left(\mathcal{Y}(\mathcal{E}) - W_{\varphi_R}(\mathcal{E})\varphi'_R(\mathcal{E})\right)\right\|}{\left\|\sum^{-1/2}\mathcal{Y}(\mathcal{E})\right\|} \qquad \textbf{15}$$

En esta expresión $W_{\varphi_R}(\mathcal{E})$ representa la matriz de observación del modelo de parámetros reducidos φ_R, usando el conjunto de datos $\mathcal{E}$. Los parámetros $\varphi'_R(\mathcal{E})$ son los valores numéricos para los parámetros base generalizados, definidos en la ec.**14** para los parámetros φ_R del modelo usando los datos $\mathcal{E}$. La matriz de ponderación $\sum^{-1/2}$ es definida como:

$$\sum^{-1/2} = diag\left(diag\left(nom(\tau_z)\right), \dots, diag\left(nom(\tau_z)\right)\right) \qquad \textbf{16}$$

Donde $nom(\tau_z)$ representa el vector de valores característicos para los elementos del vector τ_z. En general, los valores característicos basados en la estimación de datos puede ser usada, sin embargo ellos pueden ser establecidos por otras medias.

Es importante notar que para la función $\epsilon_{\tau_z}(\varphi_R, \mathcal{E})$, el argumento φ_R se refiere al conjunto de parámetros reducidos y no a sus valores numéricos. Obviamente, la computación de esta función requiere la determinación de los valores numéricos de $\varphi'_R(\mathcal{E})$.

Los índices para caracterizar la relevancia de la contribución de las diferentes fuerzas dinámicas (Coriolis, centrípetas, etc.) para la IDM han sido propuestos por [19] pero estos no se ajustan al problema de reducción de parámetros en el cual este trabajo se enfoca.

La medida del error $\epsilon_{\tau_z}(\varphi_R, \mathcal{E})$, es usada como el criterio de ajustes para la elección de los modelos de parámetros reducidos y para su validación. Por ejemplo, si el número de parámetros de n_{φ_R} del modelo deseado φ_R es fijo, el mejor modelo puede ser determinado como:

$$\varphi_R = \arg \min_{\varphi_R | n_{\varphi R} = card(\varphi_R)} \epsilon_{\tau_z}(\varphi_R, \mathcal{E}) \quad \textbf{17}$$

Donde $card(\varphi_R)$ es la cardinalidad del conjunto de parámetros φ_R.Como antes, es necesario notar que el argumento φ_R se refiere al conjunto de parámetros elegidos y no a los de sus valores numéricos. Para un deseado nivel de error,$\epsilon_{\tau_z}(\varphi_R, \mathcal{E})$, el modelo reducido es considerado aceptable si la validación y la estimación de los datos dan un nivel similar de error $\epsilon_{\tau_z}(\varphi_R, \mathrm{v}) \approx \epsilon_{\tau_z}(\varphi_R, \mathcal{E})$.

El problema con este enfoque es el número de posibles conjuntos de parámetros a ser probados para encontrar el mejor modelo de parámetros reducido con $card(\varphi_R) = n_{\varphi_R}$es alta

$$\binom{n_\varphi}{n_{\varphi_R}} = \frac{n_\varphi!}{n_{\varphi_R}!\left(n_\varphi - n_{\varphi_R}\right)!} \quad \textbf{18}$$

Si todos los conjuntos de parámetros con un número arbitrario de parámetros $n_{\varphi_R} = 1, \ldots, n_\varphi$ son probados, una mayor cantidad de candidatos es obtenida 2^{n_φ}. En general, esto excluye al enfoque de ensayo y error para el problema de reducción de parámetros. Por lo tanto, algoritmos que tengan un compromiso entre el costo computacional de búsqueda y la calidad del modelo reducido es requerida.

En [2] se establecen distintos métodos para resolver este problema entre ellos están:

1. QR Selection
2. Backward Elimination
3. Foward Selection
4. Foward Selection with two iterations (con dos iteraciones).

3.4 La selección QR

La descomposición QR de la matriz de observación $\boldsymbol{W}(\mathcal{E})$ toma la siguiente forma:

$$WE = QR \tag{19}$$

Donde la matriz WE es una columna de permutación de $\boldsymbol{W}(\mathcal{E})$, $\boldsymbol{Q}$ es una matriz ortonormal, y R es una, matriz triangular superior con los elementos de la diagonal en magnitud decreciente.

$$[W_{\varphi_R} \quad W_{\varphi_E}] = [Q_R \quad Q_E]\begin{bmatrix} R_{rr} & R_{re} \\ 0 & R_{ee} \end{bmatrix} \tag{20}$$

Donde $\boldsymbol{W_{\varphi_R}}$ contiene las primeras columnas de $\boldsymbol{WE}$, y $\boldsymbol{R_{rr}}$ es una matrix regular de $r \times r$. El modelo reducido es definido por el conjunto de parámetros asociados con las columnas de $\boldsymbol{W_{\varphi_R}}$. De la ecuación previa se tiene que:

$$W_{\varphi_R} = Q_R R_{rr}\ , y\ \ W_{\varphi_E} = Q_R R_{re} + Q_E R_{ee} \tag{21}$$

La matriz W_{φ_E} puede ser aproximada como:

$$W_{\varphi_E} \approx\ W_{\varphi_R} {W_{\varphi_R}}^{+} W_{\varphi_E} = Q_R R_{re} +\ Q_E R_{ee} \tag{22}$$

Donde:

$$\beta_{\varphi_R} = {W_{\varphi_R}}^{+} W_{\varphi_E} =\ R_{rr}^{-1} R_{re} \tag{23}$$

La ecuación **23** es una muy buena aproximación de W_{φ_E} para un conjunto de r parámetros. Notar que R_{ee} es escogido por el algoritmo QR para tener los elementos de la diagonal más pequeños y, por lo tanto, el error en la aproximación de W_{φ_E} , $Q_E R_{ee}$, es casi tan pequeña como es posible. Las ideas presentadas aquí son parcialmente inspiradas el algoritmo descrito en [20] , aunque las ideas presentadas allí se enfocan en el caso de $r = n_{\varphi_R} = rank(W(\mathcal{E}))$. Un subconjunto de algoritmos de selección con algún parecido al presentado aquí es referido como Orthogonal least Squares, descrita en [21].

El error de predicción normalizado $\epsilon_{\tau_z}(\varphi_R, \mathcal{E})$ correspondiente para los parámetros asociados con la primeras r columnas de WE en la ec **15**. Dada una tolerancia aceptable de error, y comenzando con $n_{\varphi_R} = rank\big(W(\mathcal{E})\big)$, disminuimos n_{φ_R} mientras $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) < Tol$. Entonces la parametrización más simple basada en el QR es la correspondiente a los parámetros asociados con la primeras filas de n_{φ_R} de WE**.**

3.5 La eliminación regresiva o backward elimination (BE)

El algoritmo de eliminación (BE) es iterativo y comienza usando el conjunto entero de parámetros para los modelos actuales de parámetros reducidos $\varphi_R = \varphi$. En cada iteración, el parámetro con la menor contribución significativa al error normalizado es eliminado del conjunto de parámetros del modelo actual, $\varphi_R = \varphi \setminus \varphi_e$, hasta que el criterio de error sea mayor que la tolerancia seleccionada $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) > Tol$. Esta situación es mostrada en el pseudocódigo del Algoritmo 1 usando un estilo algorítmico más formal.

$$\varphi_e = \underset{\varphi_i | \varphi_i \in \varphi_R}{\arg\min}\, \epsilon_{\tau_z}(\varphi_R \setminus \varphi_i, \mathcal{E}), \qquad \textbf{24}$$

Básicamente los parámetros que hacen una contribución más pequeña al error del modelo son quitados uno a uno. El algoritmo sería óptimo si las contribuciones al error de los diferentes parámetros φ fueran independientes. El problema es que habitualmente no lo son, y por lo tanto, eliminar un parámetro dado tiene un efecto en el error de los parámetros que quedan para el modelo reducido resultante. De esta forma, el algoritmo resultante, podría ser conveniente y es probable que funcione bien sin embargo, podemos saber de antemano que este no es óptimo.

De acuerdo con [2] una característica importante de este algoritmo es que agregado a $W(\mathcal{E})$**,** el vector $\chi(\mathcal{E})$ tiene un efecto en el orden en el cual los parámetros son reducidos. Esta información no es considerada por el algoritmo QR por lo que es una ventaja competitiva de este procedimiento.

Algoritmo 1 Backward Elimination (BE)

$\varphi_R = \varphi$
Repeat
$\varphi_e = \underset{\varphi_i | \varphi_i \in \varphi_R}{\arg\min}\, \epsilon_{\tau_z}(\varphi_R \setminus \varphi_i, \mathcal{E}),$
$\varphi_R = \varphi_R \setminus \varphi_e$
Until $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) > tol$

3.6 La selección progresiva o forward selection (FS)

El algoritmo iterativo de Forward Selection (FS) es, de alguna manera, el procedimiento inverso del anterior. Este parte de un conjunto de parámetros reducidos vacío para el modelo actual $\varphi_R = \emptyset$. En cada iteración, el parámetro con la contribución más significativa al error normalizado es agregado al conjunto de parámetros $\varphi_R = \varphi_R \cup \varphi_s$, hasta llegar a la tolerancia objetivo requerida $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) < tol$,. Esta situación es presentada de forma similar al procedimiento anterior con el Algoritmo 2:

Algoritmo 2 Foward Selection (FS)

$\varphi_R = \emptyset$
Repeat
$\varphi_e = \arg\min_{\varphi_i | \varphi_i \notin \varphi_R} \epsilon_{\tau_z}(\varphi_R \cup \varphi_i, \mathcal{E}),$
$\varphi_R = \varphi_R \cup \varphi_s$
Until $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) < tol$

En esencia, los parámetros que dan una mayor contribución al error del modelo son introducidos uno a uno, Igualmente que en el caso anterior las contribuciones no son generalmente independientes, por lo que el agregar un parámetro tiene un efecto en el error del modelo reducido resultante, Por lo tanto, es posible que el algoritmo se comporte bien sin embargo este no es óptimo.

Más adelante se comprobará que el FS trabaja ligeramente mejor que el BE. En un principio el modelo puede ser expresado exactamente en términos del conjunto de parámetros base. Como este conjunto no es único, existen muchas posibilidades de quitar los parámetros que no tengan efectos en el error. Por lo tanto, no existe un criterio para diferenciar entre las posibles eliminaciones de los parámetros pero aun, la elección tiene un efecto en el desempeño del modelo de parámetros reducidos resultante.

3.7 Selección progresiva con 2 iteraciones o Forward Selection with two iterations:

Los algoritmos presentados en las secciones anteriores aplicados son típicos en el contexto de selección de modelos. [22] [21]. La literatura sugiere la posibilidad de usar iteraciones en los algoritmos. Típicamente, una o más iteraciones de búsquedas son introducidas además de la inicial.

Como ejemplo, en este caso podemos pensar en tomar el algoritmo de FS y adaptarle una iteración adicional que podría ser leída como: dejar φ_R ser el conjunto de parámetros el inicio de una iteración dada, y establecer φ_l como el conjunto de parámetros a agregar a este conjunto en la iteración previa. El

algoritmo comienza con un vector de parámetros vacío para el modelo de parámetros reducidos $\varphi_R = \emptyset$, de igual forma, un vector extra vacío de parámetros es agregado $\varphi_l = \emptyset$, como el hecho en la iteración previa. En cada iteración, como antes se escogerá el parámetro con la contribución más significativa al error normalizado. $\varphi_R = \varphi_R \cup \varphi_s$,. Después el parámetro agregado en la iteración previa φ_l es suprimido, $\varphi_R = \varphi_R \backslash \varphi_l$ .El parámetro con la contribución al error más significativa al error es buscado nuevamente y es agregado al conjunto $\varphi_R = \varphi_R \cup \varphi_s$, Este último es comparado con el error del conjunto de parámetros anterior. Seleccionando el conjunto de parámetros que menor error introduce al modelo. Esto se repite, hasta llegar a una tolerancia $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) < tol$.
Teniendo:

$$\boldsymbol{\varphi_s = \underset{\varphi_i | \varphi_i \notin \varphi_R}{\arg\min}\, \epsilon_{\tau_z}(\varphi_R \cup \varphi_i, \mathcal{E}),} \qquad \textbf{25}$$

Esto es presentado más formalmente en el siguiente pseudocódigo correspondiente al Algoritmo 3:

Algoritmo 3 Foward Selection with two iterations (FS)

$\varphi_R = \emptyset$
$\varphi_l = \emptyset$
Repeat
$\varphi_s = \underset{\varphi_i | \varphi_i \notin \varphi_R}{\arg\min}\, \epsilon_{\tau_z}(\varphi_R \cup \varphi_i, \mathcal{E}),$
$\varphi_R = \varphi_R \cup \varphi_s$
$\varphi_l = \varphi_s$
$\varphi_{R'} = \varphi_R \setminus \varphi_l$
$\varphi_{s'} = \underset{\varphi_i | \varphi_i \notin \varphi_R}{\arg\min}\, \epsilon_{\tau_z}(\varphi_{R'} \cup \varphi_i, \mathcal{E}),$
$\varphi_{R'} = \varphi_R \cup \varphi_{s'}$
if $\epsilon_{\tau_z}(\varphi_{R'}) < \epsilon_{\tau_z}(\varphi_R) \rightarrow \varphi_l = \varphi_{s'}, \varphi_R = \varphi_{R'}$
Until $\epsilon_{\tau_z}(\varphi_R, \mathcal{E}) < tol$

Durante la implementación, el método tal cual estuvo planteado en [2]. Era incompleto, de hecho en este trabajo el método FS con dos iteraciones es modificado y completado. Los resultados y análisis serán presentados más adelante. Este método, establece soluciones que no necesariamente contiene los parametros de la solución anterior, sin embargo es posible hallar el error para un determinado número de parámetros.

Todos estos métodos anteriormente descritos fueron presentados en [2]. El presente trabajo de fin de estudios es una continuación de la investigación en la reducción de modelos dinámicos. Para este trabajo, se formularon distintas hipótesis para la mejora de estos algoritmos, sin embargo, las más relevantes y de las que se tuvieron resultados interesantes fueron las siguientes:

1. Reducción por parámetros esenciales.
2. Reducción por LASSO
3. Selección bajo la norma L1
4. Selección bajo la norma L1 con dos iteraciones.

3.8 Reduccion por parametros esenciales o Reduction by SVD Analysis (Singular Value Decomposition)

El método de reducción es mencionado con detalles en [9], este consiste en líneas generales, determinar los parámetros esenciales a partir de los parámetros base eliminando los que se consideran insignificantes mediante un análisis de SVD (Singular Value Decomposition) del cual se halla el modelo ortogonal equivalente, llevando a 0 los valores singulares.

Según [9] partimos de la suposición de que la matriz de observación W de $(rxnb)$ es de rango completo como resultado de los parámetros base y la trayectorias de excitación. Por tanto, esta puede ser descompuesta de la siguiente forma usando la descomposición en valores singulares (SVD Analysis).

En [23] se establece como SVD como una factorización de una matriz real o compleja. Es la generalización de la descomposición de valores propios de una matriz normal definida positiva para cualquier matriz $m \times n$ a través de una extensión de la descomposición polar.

Formalmente, la descomposición en valores singulares de una matriz M real o compleja de $m \times n$ es una factorización de la forma $U\Sigma V^*$, donde U es una matriz $m \times m$ unitaria real o compleja, Σ es una matriz $m \times n$ diagonal rectangular con números reales no-negativos en la diagonal y V es una matriz unitaria de $n \times n$ real o compleja. Los valores de la diagonal de Σ son conocidos como los valores singulares de M. Las columnas de U y las columnas de V son denominadas como vectores singulares de izquierdos y vectores singulares derechos de M

$$W = USV^T \tag{26}$$

$$W = [U_1 \quad U_2]\begin{bmatrix}\Sigma \\ 0_{(r-nb)xnb}\end{bmatrix}V^T = U_1\Sigma V^T \tag{27}$$

Donde:

$U = [U_1 \quad U_2]$, y V son las matrices ortogonales (rxr) y $(nbxnb)$ respectivas.

$\Sigma = diag(\sigma_1, \dots, \sigma_{nb})$ Con $\sigma_1 \geq \cdots \geq \sigma_{nb}$

Usando **27** el sistema compatible equivalente se convierte en:

$$\tau_z = U_1 \Sigma V^T X \quad \textbf{28}$$

Lo que nos permite definir el sistema equivalente como:

$$G = {U_1}^T Y \quad \textbf{29}$$

$$Z = V^T X \quad \textbf{30}$$

Por lo que el sistema ortogonal equivalente queda redefinido como:

$$G = \Sigma Z \quad \textbf{31}$$

La reducción por descomposición de valores singulares parte en forzar algunos valores de **Σ** a 0 el interés de esta transformación es que la matriz **Σ** es una matriz diagonal, por lo que los efectos de la reducción del modelo en el error y en la condición numérica son dados explícitamente.

Por lo que definimos la Matriz P^T la cual llamaremos matriz de permutación y su función en líneas generales es la de organizar los componentes de la matriz G en orden decreciente $G' = P^T G$. Por lo que $|{G'}_i| \geq |{G'}_i + 1|$. El sistema ortogonal equivalente ec.**31** después de la permutación y la separación en dos grupos ${G'}_1$y ${G'}_2$ pasa a ser:

$$G' = \Sigma' Z' \quad \textbf{32}$$

Con $G' = \begin{bmatrix} {G'}_1 \\ {G'}_2 \end{bmatrix}, \Sigma' = P^T \Sigma P = \begin{bmatrix} {\Sigma'}_1 & 0 \\ 0 & {\Sigma'}_2 \end{bmatrix}, \; Z' = P^T Z = \begin{bmatrix} {Z'}_1 \\ {Z'}_2 \end{bmatrix}$

Para este caso valor de ${G'}_2$ será despreciado y su cálculo se explicara más adelante.

Por lo que la reducción del modelo equivalente ec.**32** consiste en considerar:

$$\widehat{G'} = {\Sigma'}_r \widehat{Z'} \quad \textbf{33}$$

Donde:

$${\Sigma'}_r = \begin{bmatrix} {\Sigma'}_1 & 0 \\ 0 & 0 \end{bmatrix} \quad 34$$

La solución de $G' = \Sigma'_r z' + b$ la cual es $\hat{Z} = \arg\min_{z'} \|G' - \Sigma'_r z'\|$, es dada por:

$$\hat{Z}' = \begin{bmatrix} \Sigma'^{-1}_1 G'_1 \\ \hat{Z}'_2 \end{bmatrix} \quad 35$$

Donde $\hat{\boldsymbol{Z}}'_{\boldsymbol{2}}$ puede ser escogido arbitrariamente (esto es explicado con mayor detalle en [9]) por lo que el modelo simplificado quedaría expresado como:

$$\hat{G}' = \begin{bmatrix} G'_1 \\ 0 \end{bmatrix} \quad 36$$

Una vez simplificado tenemos que regresar el cambio del modelo ortogonal equivalente al modelo inicial haciendo:

$$\hat{Y} = W_s \hat{X} \quad 37$$

Con $W_s = P\Sigma'_r P^T V^T$ y $\hat{X} = VP\hat{Z}'$

La norma mínima del error del modelo queda definida como:

$$\left\| Y - \hat{Y} \right\| = \|G'_2\| \quad 38$$

Ya que $\|G'_2\|$ representa el error del modelo, Esta debe ser escogida como la más pequeña posible.

En este método es importante resaltar entre otras cosas que no existe relación analítica entre la matriz de observación W_sy W **. A efectos, la obtencion de los parametros, es detallada en** [20]**, sin embargo las soluciones obtenidas número de parametros n, igual que en el caso del FS iterativo no estarán contenidas en N+1. Por tanto, los parametros escogidos para cada iteración no se tomaran en cuenta para este trabajo.**

.

3.9 Reducción mediante LASSO.

Según [24] [25] LASSO (least absolute shrinkage and selection operator, por sus siglas en inglés), es un método de análisis de regresión que realiza selección de variables y regularización para mejorar la exactitud e interpretación del modelo estadístico producido por este. Lasso fue formulado originalmente para el método de mínimos cuadrados y este caso simple revela una cantidad substancial acerca del comportamiento del estimador, incluyendo su relación con Ridge regression y selección de subconjuntos (de variables) y la conexión entre los coeficientes estimados con LASSO y el llamado 'soft thresholding'. También

revela que (al igual que la Regresión Lineal estándar) los coeficientes estimados no necesariamente son únicos si las variables independientes son colineales.

Ridge regression o Regularización de Tikhonov [26] [27] es el método más comúnmente utilizado para la regularización de problemas. Según [28] En estadística la regularización se define como el proceso por el cual se introduce información adicional resolver problemas condicionados, no bien definidos, o para prevenir sobreajuste.

Si suponemos que para una matriz conocida A y un vector b nos interesa encontrar un vector x tal que:

$$Ax = b$$

El enfoque usual es la solución de mínimos cuadrados. Sin embargo, si x no satisface la ecuación o más de un vector x lo hace, se considera que la solución no es única por lo que se considera que el problema no está bien definido o está condicionado. En tales casos la solución de mínimos cuadrados lleva a una solución sobrejaustada o sub-ajustada. La solución de mínima norma se basa en minimizar la suma de los cuadrados de los residuos que puede ser expresado como:

$$\|A_x - b\|_2^2$$

Por tanto, para dar preferencia a una solución en particular con unas propiedades deseadas, es agregado un término de regularización que puede ser expresado como:

$$\|A_x - b\|_2^2 + \|\Gamma x\|_2^2$$

Convenientemente el término Γ es definido como la matriz de Tikhonov. En muchos casos esta es escogida como un múltiplo de la matriz de la identidad $(\Gamma = \alpha \mathrm{I})$, dando preferencia a soluciones con la norma más pequeña, esto es conocido como la Regularización bajo la norma L2 o Ridge regresión.

En el caso de LASSO, usando la notación de [29], para un valor dado de λ no-negativo, LASSO resuelve el problema:

$$\min_{x_0 x} \left(\frac{1}{2N} \sum_{i=1}^{N} (b - x_0 - x A_i^T x)^2 + \lambda \sum_{J=1}^{P} |x_j| \right) \quad (39)$$

Donde:

- N es el número de observaciones.
- b_i es la respuesta a la observación i.

- A_i son los datos, un vector de longitud P en la observación i.
- λ es un parámetro no-negativo de regularización correspondiente a un valor de lambda.
- Los parámetros x_0 y x son un escalar y un vector de longitud P respectivamente. A medida de que λ incrementa, el número de componentes no-zeros de x disminuye. El problema de LASSO involucra a la Norma L^1de x.

La aplicación de LASSO en este contexto resulto complicada de establecer, si bien es cierto que es capaz de encontrar diferentes combinaciones de parámetros que lleguen a encontrar el error mínimo, el problema principal residía en que el número de parámetros que LASSO escogerá para satisfacer ese error, no eran mínimos, de hecho terminaba siendo una cantidad aleatoria de parámetros, que satisfacen ese error y de los cuales no se podía dar relación con respecto a otros métodos, por lo que a pesar de que tiene sentido su uso en este contexto, esta solo será mencionada. Su implementación y resultados quedaran fuera de los objetivos de este trabajo.

3.10 La norma de un vector

Según [30] Un vector es un elemento de un espacio vectorial del que, en ocasiones, especialmente en física y geometría, interesa conocer su longitud. Para ello, se hace necesario definir un operador norma que determine la longitud o magnitud del vector bajo consideración ya que este acto, pese a lo que pudiéramos creer, no es un problema trivial; especialmente desde la aparición de las geometrías no euclídeas para las que surge.

La definición general de norma se basa en generalizar a espacios vectoriales abstractos la noción de módulo de un vector de un espacio euclídeo. Recuérdese que en un espacio no euclídeo el concepto de camino más corto entre dos puntos ya no es identificable necesariamente con el de la línea recta; por ello, se utilizan las propiedades operacionales de la norma euclídea definida más abajo para extraer las condiciones que debe cumplir la "longitud de un vector", o norma vectorial, en un espacio vectorial cualquiera. Estas condiciones básicas son:

- Siempre es no negativa e independiente del sentido (orientación) de la medición.
- La longitud debe ser directamente proporcional al tamaño (es decir, doble -o triple- de tamaño significa doble -o triple- de longitud).
- La longitud entre dos puntos será siempre menor o igual que la suma de longitudes desde esos mismos dos puntos a un tercero diferente de ellos (desigualdad triangular: la suma de dos lados de un triángulo nunca

es menor que el tercer lado, también generalizada en la desigualdad de Cauchy-Schwartz). Se presentan dos maneras de forma, una casi directa y apunta a lo dicho: longitud de vector. La otra usa la noción de operador y mayor simbolismo de la matemática formal (tipo Bourbaki)

3.10.1 Norma L1

Uno de los principios centrales del procesado de señales es la teoría de muestreo de Shannon/Nyquist [31] [32] en la cual se establece que el número de muestras necesarias para capturar una señal esta dictada por su ancho de banda. Muy recientemente, ha emergido una teoría alternativa de "muestreo comprimido". Usando algoritmos no lineales de recuperación (basado en la optimización convexa), de señales e imágenes pueden ser reconstruidas de lo que aparentan ser datos bastante incompletos. El muestreo comprimido nos muestra como la compresión de los datos puede ser implícitamente incorporada en el proceso de adquisición de datos, dándonos un nuevo punto de partida para un diverso cumulo de aplicaciones, incluyendo imágenes tomográficas, conversiones analógico-digitales y fotografía digital.

En líneas generales se establece que un vector disperso $X_0 \in \mathbb{R}^N$ Puede ser recuperado de un pequeño número de medidas lineales A usando $b = Ax_0 \in \mathbb{R}^K, K \ll N$ (o ó $b = Ax_0 + e$ cuando hay ruido remanente) resolviéndolo como un problema convexo.

Según [1] nuestro problema de interés se establece como:

- Min-L1 con restricciones de igualdad (equality constrains):

$$\min\|x\|_1 \quad sujeto\ a \quad Ax = b, \tag{40}$$

También conocido como "basis pursuit" o búsqueda fundamental, el cual busca el vector con la norma L1 más pequeña que exprese las observaciones en b. La norma L1 se expresa simplemente como la suma de los valores absolutos de las columnas:

$$\|x\|_1 := \sum_i |x_i| \tag{41}$$

Según [33] [34] se establece que, si una solución suficientemente dispersa (sparse) de x_0 Tal que $Ax_0 = b$ existe. Entonces el problema planteado en la ec.**40** será capaz de encontrarla.

La norma L1 es un método de optimización en el cual se plantea resolver sistemas de ecuaciones distintos a los metodos usados convencionalmente, usando algoritmos convexos, logran resolver sistemas de una manera distinta. Un ejemplo claro de esto es expresado en el siguiente problema:

-Hallar dos números cuyo promedio es 3:

El problema puede ser planteado como un sistema de ecuaciones de la siguiente manera:

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix}, b = 3$$

Por tanto:

$$Ax = b$$

Donde x será el vector que satisface la ecuación. Donde, sí hallamos la solución de norma mínima tenemos:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

Sin embargo los valores de x pueden tomar otras soluciones que satisfacen el problema en este caso:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

Ambas soluciones son válidas sin embargo en una de ellas una de sus componentes es 0.

El uso de esta técnica proviene de la restauración de imágenes, en donde se obtiene una señal con millones de valores de pixeles, el trabajo es re-generar esa señal a la original, donde tendríamos que comprimir la muestra b y resolver Ax=b. hay un gran número de soluciones posibles, las bases para escoger la correcta involucra a la norma L1 expresada en la ec.**41**.

La clave para el comprimir estas señales viene de los elementos dispersos de la norma L1. Si la expansión de la imagen original o señal como una combinación de las funciones bases tiene muchos coeficientes 0, entonces es posible reconstruir la señal exacta. En principio, computar esta reconstrucción debería involucrar un gran conjunto de valores lo que lo hace un problema de combinatoria de números cuya complejidad computacional es impráctica. (Es NP-hard). Afortunadamente, mediante la norma L1 se puede resolver este problema ya que su implementación puede ser puesta como un problema de programación lineal y resuelto con el método del punto interior.

Para resolver estos tipos de problema existen a día de hoy diferentes métodos sin embargo de los que se van a hablar en este trabajo estarán basados en el método del punto interior para optimización convexa expuestos en [1].

3.11 Método del punto interior (Interior point methods):

Avances en los métodos del punto interior para optimización convexa en los últimos años, dirigidos por el trabajo expuesto en [35] han logrado crear integradores para estos tipos de problemas de manera factible.

3.11.1 Algoritmo primal-dual para programación lineal.

En el capítulo 11 de [36], se establece un algoritmo primal-dual relativamente simple para programación lineal el cual, se ha seguido muy de cerca en [1] para su implementación.

La forma estándar de programación lineal es:

$$\min_z \langle c_0, z\rangle \quad sujeto\ a \begin{matrix} A_0 z = b \\ f_i(z) \leq 0 \end{matrix} \tag{42}$$

Donde el vector de búsqueda $z \in \mathbb{R}^N$, $b \in \mathbb{R}^K$, A_0 es una matriz de KxN y cada una de las $f_i, i = 1, \dots, m$ es un funcional lineal:

$$f_i(z) = \langle c_i, z\rangle + d_i \tag{43}$$

Para algunos $c_i \in \mathbb{R}^N, d_i \in \mathbb{R}$. En el punto optimo z^*,Existiran vectores duales $v^* \in \mathbb{R}^K, \lambda^* \in \mathbb{R}^m, \lambda^* \geq 0$ tal que las condiciones de Karush-Kuhn-Tucker [37] sean satisfechas:

$$(KKT) \quad c_0 + A_0^T v^* + \sum_i \lambda_i^* c_i = 0, \tag{44}$$

$$\lambda_i^* f_i(z^*) = 0, i = 1, \dots, m, \tag{45}$$

$$A_0 z^* = b, \tag{46}$$

$$f_i(z^*) \leq 0, i = 1, \dots, m, \tag{47}$$

En resumen, el algoritmo primal-dual encuentra el óptimo de z^* (junto con los vectores óptimos duales v^* y λ^*) al resolver este sistema de ecuaciones no lineales. El procedimiento de solución es el método clásico de Newton: el

método del punto interior (z^k, v^k, λ^k) (para el cual se establece $f_i(z^k) < 0, \lambda^k > 0$), el sistema es linealizado y resuelto. Sin embargo el paso para un nuevo punto $(z^{k+1}, v^{k+1}, \lambda^{k+1})$ debe ser modificado para que se mantenga en el interior de la solución.

En la práctica, se suele relajar las condiciones de holgura complementaria $\lambda_i f_i = 0$ a:

$$\lambda_i^k f_i(z^k) = -1/\tau^k, \tag{48}$$

En donde juiciosamente se incrementa el parámetro τ^k a medida que se va avanzando a través de las iteraciones de Newton. Esto condiciona la solución de ecuaciones linealizadas hacia el interior permitiendo un suave y bien definido "camino central" desde un punto interior para la solución en el contorno.

Los residuos de las condiciones primarias duales y centrales cuantifican que tan cerca está un punto (z, v, λ) de satisfacer las condiciones de Karun-Kush-Tucker. Con la ec.**48** en el lugar de la condiciones de holgura:

$$r_{dual} = c_0 + A_0^T v + \sum_i \lambda_i c_i \tag{49}$$

$$r_{cent} = -\Lambda f - (1/\tau)1 \tag{50}$$

$$r_{pri} = A_0 z - b \tag{51}$$

Donde Λ es una matriz diagonal con $(\Lambda)_{ii} = \lambda_i$, y $f = \left(f_1(z) \dots f_m(z)\right)^T$.

Desde un punto (z, v, λ), se busca encontrar un paso $(\Delta z, \Delta v, \Delta\lambda)$ tal que:

$$r_\tau(z + \Delta z, v + \Delta v, \lambda + \Delta\lambda) = 0 \tag{52}$$

Usando la expansión de Taylor alrededor de (z, v, λ), tenemos:

$$r_\tau(z + \Delta z, v + \Delta v, \lambda + \Delta\lambda) \approx r_\tau(z, v, \lambda) + J_{r_\tau}(z, v, \lambda)\begin{pmatrix}\Delta z \\ \Delta v \\ \Delta\lambda\end{pmatrix} \tag{53}$$

Donde $J_{r_\tau}(z, v, \lambda)$ es el jacobiano de r_τ por lo que tenemos el siguiente sistema:

$$\begin{pmatrix} 0 & A_0^T & C^T \\ -\Lambda C & 0 & -F \\ A_0 & 0 & 0 \end{pmatrix}\begin{pmatrix}\Delta z \\ \Delta v \\ \Delta\lambda\end{pmatrix} = -\begin{pmatrix} c_0 + A_0^T v + \sum_i \lambda_i c_i \\ -\Lambda f - (1/\tau)1 \\ A_0 z - b \end{pmatrix} \tag{54}$$

Donde la matriz C de longitud $m \times N$ tiene el vector c_i^T como columnas, y F es diagonal siendo $(F)_{ii} = f_i(z)$. Es posible obtener $\Delta\lambda$ usando:

$$\Delta\lambda = -\Lambda F^{-1} C\Delta z - \lambda - (1/\tau) f^{-1} \quad \textbf{55}$$

Por lo que el sistema nos queda como:

$$\begin{pmatrix} -C^T F^{-1}\Lambda C & A_0^T \\ A_0 & 0 \end{pmatrix}\begin{pmatrix} \Delta z \\ \Delta v \end{pmatrix} = \begin{pmatrix} -c_0 + (1/\tau)C^T f^{-1} - A_0^T v \\ b - A_0 z \end{pmatrix} \quad \textbf{56}$$

Con $(\Delta z, \Delta v, \Delta\lambda)$ tenemos la dirección del paso de solución. Sim embargo para escoger la longitud del paso $0 < s \leq 1$ hay que preguntarse si se satisfacen dos criterios:

1. Si $z + s\Delta z$ y $\lambda + s\Delta\lambda$ están en el interior, por ejemplo $f_i(z + s\Delta z) < 0, \lambda_i > 0$ para todo i.
2. Si la norma de los residuales han disminuido suficientemente cuando:

$$\|r_t(z + s\Delta z, v + s\Delta v, \lambda + s\Delta\lambda)\|_2 \leq \|(1 - \alpha s) \cdot r_t(z, v, \lambda)\|_2 \quad \textbf{57}$$

Donde α es un valor especificado por el usuario y se suele establecer como 0.01.

Por lo que se suele escoger el paso máximo que nos mantiene en el interior estableciendo:

$$I_f^+ = \{i : \langle c_i, \Delta z\rangle > 0\}, \quad I_\lambda^- \{i : \Delta\lambda < 0\} \quad \textbf{58}$$

Ajustando:

$$s_{max} = 0.99 \cdot \min\left\{1, \{-f_i(z)/\langle c_i, \Delta z\rangle, i \in I_f^+\}, \{-\lambda_i/\Delta\lambda_i, i \in I_\lambda^-\}\right\} \quad \textbf{59}$$

Por tanto comenzando con $s = s_{max}$, se revisa que los criterios anteriormente comentado se satisfagan; si este no es el caso, se ajusta $s' = \beta \cdot s$ y se intenta nuevamente. Se tiene pre establecido el valor de $\beta = 1/2$.

Cuando el r_{dual} y r_{pri} son pequeños la brecha de la dualidad $\eta = -f^T\lambda$ es una aproximación de que tan cerca de cierto punto (z, v, λ), está de ser óptimo. El algoritmo primal-dual repite las iteraciones descritas anteriormente hasta que el valor de la brecha η haya disminuido hasta una tolerancia dada.

Cuando N y K son muy grandes, formar la matriz y luego resolver el sistema lineal de ecuaciones de la ec. **56**. No es factible. Sin embargo, es posible usar integradores como el método de gradientes conjugados [38] [39]. El método de gradientes conjugados es iterativo requiriendo unos cientos aplicaciones de las matrices de restricciones (en términos generales) para obtener una solución

precisa. El integrador de gradientes conjugados usado en este trabajo viene de [1] basado en [38] [39].

En líneas generales, lo explicado anteriormente sirve como una breve introducción al cálculo de optimización convexa, sin embargo, el algoritmo aplicado para la resolución de la ec **40**. Está basado en la explicación dada en el Anexo A de [1] donde se establece lo siguiente:

Partimos de:

$$\min\|x\|_1 \quad sujeto\ a \quad Ax = b,$$

Cuando x, A y b son reales el problema puede ser replanteado como un problema de programación lineal de la siguiente manera:

$$\min_{x,u} \sum_i u_i \; sujeto\ a \quad \begin{matrix} x_i - u_i \leq 0 \\ -x_i - u_i \leq 0 \\ Ax = b \end{matrix} \tag{60}$$

El cual puede ser resuelto usando el algoritmo estándar primal-dual explicado anteriormente por lo que denominamos:

$$f_{u1;i} \coloneqq x_i - u_i \tag{61}$$

$$f_{u2;i} \coloneqq -x_i - u_i \tag{62}$$

Con $\lambda_{u1;i}, \lambda_{u2;i}$, las correspondientes variables duales y dejando que f_{u1} sea el vector $\left(f_{u1;1} \ \dots \ f_{u1;N}\right)^T$ y de misma manera para los vectores $f_{u2}, \lambda_{u1}, \lambda_{u2}$. Notar que:

$$\nabla f_{u1;i} = \begin{pmatrix} \delta_i \\ -\delta_i \end{pmatrix}, \nabla f_{u2;i} = \begin{pmatrix} -\delta_i \\ -\delta_i \end{pmatrix}, \nabla^2 f_{u1;i} = 0, \nabla^2 f_{u2;i} = 0 \tag{63}$$

Donde δ_i es el vector de base estándar para el componente i, por lo tanto en un punto $(x, u; v, \lambda_{u1}, \lambda_{u2})$. De este modo, los residuos centrales y duales son:

$$r_{Cent} = \begin{pmatrix} -\Lambda_{u1} f_{u1} \\ -\Lambda_{u2} f_{u2} \end{pmatrix} - (1/\tau)1 \tag{64}$$

$$r_{dual} = \begin{pmatrix} \lambda_{u1} - \lambda_{u2} + A^T v \\ 1 - \lambda_{u1} - \lambda_{u2} \end{pmatrix} \tag{65}$$

Por tanto el sistema de ecuaciones queda de la siguiente manera:

$$\begin{pmatrix} \Sigma_1 & \Sigma_2 & A^T \\ \Sigma_2 & \Sigma_1 & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta v \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} := \begin{pmatrix} (-1/\tau) \cdot (-f_{u1}^{-1} + f_{u2}^{-1}) - A^T v \\ -1 - (1/\tau) \cdot (f_{u1}^{-1} + f_{u2}^{-1}) \\ b - Ax \end{pmatrix} \quad 66$$

Con:

$$\Sigma_1 = \Lambda_{u1} F_{u1}^{-1} - \Lambda_{u2} F_{u2}^{-1}, \quad \Sigma_2 = \Lambda_{u1} F_{u1}^{-1} + \Lambda_{u2} F_{u2}^{-1} \quad 67$$

F son matrices diagonales con $(F_\bullet)_{ii} = f_{\bullet;i}$, y $f_{\bullet;\mathrm{i}}^{-1} = 1/f_{\bullet;i}$. Estableciendo:

$$\Sigma_x = \Sigma_1 - \Sigma_2^2 \Sigma_1^{-1} \quad 68$$

Despejando:

$$\Delta x = \Sigma_x^{-1}(w_1 - \Sigma_2 \Sigma_1^{-1} w_2 - A^T \Delta v) \quad 69$$

$$\Delta \mathrm{u} = \Sigma_1^{-1}(w_2 - \Sigma_2 \Delta \mathrm{x}) \quad 70$$

Resolviendo:

$$-A\Sigma_x^{-1} A^T \Delta v = w_3 - A(\Sigma_x^{-1} w_1 - \Sigma_x^{-1} \Sigma_2 \Sigma_1^{-1} w_2) \quad 71$$

La ec. **71** es un sistema definido positivo de ecuaciones, el cual es resuelto por el método de los gradientes conjugados:

Para un dado $(\Delta x, \Delta u, \Delta v)$ Por lo que el cambio en las variables duales es computado como:

$$\Delta \lambda_{u1} = \Lambda_{u1} F_{u1}^{-1}(-\Delta x + \Delta u) - \lambda_{u1} - (1/\tau) f_{u1}^{-1} \quad 72$$

$$\Delta \lambda_{u2} = \Lambda_{u2} F_{u2}^{-1}(\Delta x + \Delta u) - \lambda_{u2} - (1/\tau) f_{u2}^{-1} \quad 73$$

3.12 Método de los gradientes conjugados

El método del gradiente conjugado es un algoritmo que sirve para resolver numéricamente los sistemas de ecuaciones cuyas matrices son simétricas y definidas positivas. Es un método iterativo así que se puede aplicar a los sistemas dispersos que son demasiado grandes para ser tratados por métodos directos. Este método también es usado para minimizar funciones cuadráticas convexas, por lo que este método se hace adecuado para resolver nuestro problema.

En [38] y [39] se tiene una explicación más profunda y bastante detallada del método, sin embargo este puede ser resumido de la siguiente manera:

Dado los valores de entrada A, b, un valor inicial de x, el máximo número de iteraciones i_{max}, una tolerancia del error cualquiera $\varepsilon < 1$:

Algoritmo 5 : Gradientes conjugados

$i \Leftarrow 0$
$r \Leftarrow b - Ax$
$d \Leftarrow r$
$\delta_{new} \Leftarrow r^T r$
$\delta_0 \Leftarrow \delta_{new}$
While $i < i_{max}\ and\ \delta_{new} > \varepsilon^2 \delta_0\ do$
    $q \Leftarrow Ad$
    $\alpha \Leftarrow \dfrac{\delta_{new}}{d^T q}$
    $x \Leftarrow x + \alpha d$
    If i es divisible por 50
        $r \Leftarrow b - Ax$
    Else
        $r \Leftarrow r - \alpha q$
    $\delta_{old} \Leftarrow \delta_{new}$
    $\delta_{new} \Leftarrow r^T r$
    $\beta \Leftarrow \dfrac{\delta_{new}}{\delta_{old}}$
    $d \Leftarrow r + \beta d$
    $i \Leftarrow i + 1$

El algoritmo termina cuando el número máximo de iteraciones i_{max} han sido excedidas, o cuando $\|r_{(i)}\| < \varepsilon \|r_{(0)}\|$.

La fórmula recursiva rápida para el residuo es usualmente usada pero solo cada 50 iteraciones. El residuo exacto es recalculado para quitar errores de coma flotantes acumulados. Por supuesto, el numero 50 es arbitrario; para grandes iteraciones $n, \sqrt{n}$, puede ser más apropiado. Si la tolerancia es grande, el residuo no necesita ser corregido. Si la tolerancia está cercana a los límites de la precisión de los comas flotantes de la máquina, un ensayo debería, ser agregado después de que δ es evaluado para revisar si $\delta \leq \varepsilon^2 \delta_0$ y si este ensayo se mantiene como cierto, el residuo exacto debería ser recalculado y δ reevaluado. Esto previene que el proceso termine antes debido a un error de coma flotante.

3.13 Aplicación de la norma L1 en la reducción de parámetros

Como hemos visto en apartados anteriores la norma L1 es usada en el contexto de transmisión y compresión de imágenes o señales, para así disminuir la información o datos a transmitir, perdiendo la menor calidad de la información posible en el proceso.

Esta misma teoría aplicada en el contexto de sistemas dinámicos es la que se presenta en este trabajo para reducir los parámetros de los modelos dinámicos, en el pasado se tendría que tener un conocimiento profundo del sistema o cuanto menos aplicar los algoritmos presentados en [2] para poder eliminar parámetros de manera eficiente. El problema presentado en este caso es el mismo:

"Para un modelo dinámico dado, con un conjunto de parámetros conocidos, se busca determinar el mínimo conjunto de parámetros del modelo que logre aproximar los datos del modelo, con una deseada precisión".

Por lo que al igual que en [2] podremos partir de la ec **13**:

$$W_{\varphi_R}(\mathcal{E})\varphi'_R = X(\mathcal{E}) \qquad \textbf{74}$$

Donde $W(\mathcal{E})$ (como se ha establecido anteriormente) es la llamada matriz de observación para un conjunto de mediciones $\mathcal{E}$. φ'_R parámetros dinámicos del sistema a reducir. $X(\mathcal{E})$ Representa el vector de fuerzas externas aplicadas.

Este sistema de ecuaciones puede ser visto de manera similar como el sistema de ecuaciones base a resolver para la norma L1:

$$W_{\varphi_R}(\mathcal{E})\varphi'_R = X(\mathcal{E}) \rightarrow Ax = b \qquad \textbf{75}$$

Donde A expresaría la matriz de observación W_{φ_R}, x expresaría el vector de parámetros a reducir φ'_R y b expresaría el vector de fuerzas exteriores $X(\mathcal{E})$ para un estado $\mathcal{E}$. Por tanto la idea general es llevar el mismo principio visto anteriormente con la norma L1 y resolver el sistema de ecuaciones presentado mediante la norma L1, el cual plantea resolver un problema de optimización convexa que es expresado como:

$$\min\|x\|_1 \quad sujeto\ a \quad Ax = b, \qquad \textbf{76}$$

La intención con este método es hallar una solución dispersa en la cual muchos de los valores dentro de la matriz sean 0 o muy cercanos a 0, el algoritmo se resolverá con el método explicado en el aparatado anterior, por tanto visto

desde otro punto de vista el método implícitamente valorará que parámetros son más importantes o menos importantes para el modelo y por tanto la solución obtenida en el modelo dará magnitudes muy altas para aquellos parámetros que son más importantes y magnitudes muy bajas para los menos importante por tanto en una primera impresión tendremos que el parámetro de mayor magnitud representará el parámetro con mayor importancia dentro del modelo, de igual forma los parámetros que sean más pequeños son los parámetros que se podrían quitar del modelo.

Por tanto, el orden en el cual tendremos que quitar los parámetros del sistema, será el orden de menor a mayor de los parámetros obtenidos como solución. El método de reducción de parámetros se puede resumir de la siguiente manera:

Partimos de:

$$W_{\varphi_R}(\mathcal{E})\varphi'_R = X(\mathcal{E}); \quad (77)$$

Resolvemos

$$min \left\| \varphi'_{op} \right\|_1 \quad sujeto\ a \quad W_{\varphi_{op}}(\mathcal{E})\varphi'_{op} - X(\mathcal{E}) = 0 \quad (78)$$

Utilizamos la nomenclatura de φ'_{op} ya que la solución obtenida de este sistema no se corresponde a los parámetros reales, el punto de esto es sin más diferenciar el vector de parámetros del vector optimizado. Lo que procede ahora es ir quitando los parámetros φ'_Rdel modelo calculando el error en el orden obtenido por el vector de parámetros del vector optimizado φ'_{op} de la siguiente manera:

$$\varphi'_s = \min \left\| \varphi'_{op} \right\| \quad (79)$$

φ'_s Es un valor que expresa el parámetro a quitar del sistema dinámico por tanto tenemos que:

$$\varphi'_R = \varphi'_R \backslash \varphi'_s \quad (80)$$

Al quitar el parámetro hay que quitar la fila del vector de observaciones que está asociada al parámetro que se ha retirado en este caso por tanto la fila asociada a ese parámetro la denominaremos $W_{\varphi_s}(\mathcal{E})$ Por lo que:

$$W_{\varphi_N}(\mathcal{E}) = W_{\varphi_R}(\mathcal{E}) \backslash W_{\varphi_s}(\mathcal{E}) \quad (81)$$

Por tanto, se tiene que volver a recalcular el error y para esto tenemos que volver a hallar el valor nuevo valor de φ'_R que haga que $W_{\varphi_R}(\mathcal{E})\varphi'_R = X(\mathcal{E})$; para esto resolvemos:

$$min\left\|\varphi'_N\right\|_2 \quad sujeto\ a \quad W_{\varphi_N}(\mathcal{E})\varphi'_N - X(\mathcal{E}) = 0 \tag{82}$$

Resolver este problema nos dará un nuevo vector de parámetros reducidos.

$$X(\mathcal{E})_N = W_{\varphi_N}(\mathcal{E})\varphi'_N \tag{83}$$

Por tanto el error cometido al retirar un parámetro del modelo es calculado de la siguiente manera:

$$\varepsilon = \frac{\|X(\mathcal{E}) - X(\mathcal{E})_N\|}{\|X(\mathcal{E})\|} \tag{84}$$

A simple vista es posible ver que el método ofrece una manera bastante optima de retirar los parámetros del modelo dinámico, sin embargo, existen diversos factores que hacen que no lo sea, tales como la solución al problema presentado en la ec.**78** como hemos visto en el apartado 3.11 la solución del problema convexo en si viene con un error implícito ya que, si nos fijamos en la tercera restricción de la ec.**60** tenemos:

$$Ax = b$$

Si pensamos fríamente, hallar una solución exacta bajo la norma L1 podría ser una solución directa a nuestro problema, sin embargo esto no es más que una utopía ya que, hay 3 premisas a tomar en cuenta. La primera es que para dar con una solución exacta en un problema de optimización en ciertos casos terminan siendo difícil de hallar en un tiempo razonable por lo que es necesario recurrir a soluciones aproximadas, que a fin de cuentas es mejor que esperar hasta dar con una solución que puede ser casi igual de buena a la aproximada, para resolver nuestro problema estamos sujetos a las condiciones de Karush-Kuhn-Tucker, en las cuales se establece que un punto es óptimo si se satisfacen las siguientes condiciones:

$$(KKT) \quad c_0 + A_0^T v^* + \sum_i \lambda_i^* c_i = 0, \tag{85}$$

$$\lambda_i^* f_i(z^*) = 0, i = 1, \dots, m, \tag{86}$$

$$A_0 z^* = b, \tag{87}$$

$$f_i(z^*) \leq 0, i = 1, \dots, m, \tag{88}$$

Por lo que entre más cercano nos encontremos de esas condiciones más cercanos estaremos de una solución óptima o en otras palabras la solución a encontrar es cuasi-optima. La otra premisa importante a tomar en cuenta es el error que se comete debido a que se hacen simplificaciones o aproximaciones de valores que no son exactos antes de aplicar el método de optimización, tales como en la Ec. **53**:

$$r_\tau(z+\Delta z, v+\Delta v, \lambda+\Delta\lambda) \approx r_\tau(z,v,\lambda) + J_{r_\tau}(z,v,\lambda)\begin{pmatrix}\Delta z\\ \Delta v\\ \Delta\lambda\end{pmatrix}$$

Donde usamos la expansión de Taylor para hallar una aproximación al paso que se tiene que hacer en la dirección de la solución. O en su defecto la solución por gradientes conjugados de la expresión $Ax = b$. Por lo que en el método general primal-dual existen 2 residuos a tomar en cuenta, ya que establecen que tan buena es la solucion obtenida tales comoel residuo del algoritmo primal dual (ec **49**) y el residuo obtenido de la solución por gradientes conjugados (ec **51**) . Por último y no menos importante es posible que existan valores de parámetros fundamentales, que sin más su magnitud es pequeña pero es fundamental al modelo, que al retirar parámetros no distinguimos estos de los más importantes, simplemente se hace una distinción en magnitud dada por la norma L1.

Más adelante se presenta la implementación de la norma L1, para la reducción de parámetros en el contexto de la dinámica de sistemas multicuerpos.

3.14 Selección bajo la norma L1 Iterativo.

Como hemos podido lograr ver en el apartado anterior, el principio de la norma L1 para resolver problemas convexos, puede en la teoría, ser muy útil. Sin embargo y como veremos más adelante no es un método óptimo, comparado a los ya aplicados, llevando al desarrollo de otro método. Este método consiste en la aplicación de esta norma de manera recursiva mientras se van reduciendo los parámetros. Para ilustrarlo de una mejor manera partimos de:

$$W_{\varphi_R}(\mathcal{E})\varphi'_R = X(\mathcal{E}); \qquad \textbf{77}$$

Resolvemos

$$min \left\|\varphi'_{op}\right\|_1 \quad sujeto\ a \quad W_{\varphi_{op}}(\mathcal{E})\varphi'_{op} - X(\mathcal{E}) = 0 \qquad \textbf{78}$$

A partir de este punto en donde seleccionamos el parámetro a retirar es donde el método cambia, en el apartado anterior tomábamos la solución de φ'_{op} se listaban y se iban retirando los parámetros del modelo según el orden obtenido al listar de menor a mayor el vector φ'_{op}, este orden iba retirando de φ'_R parámetro a parámetro hasta haberlos quitados todos, en este caso lo que se dispondrá a hacer es:

a) Resolver el problema inicial :

$$min \left\|\varphi'_{op}\right\|_1 \quad sujeto\ a \quad W_{\varphi_{op}}(\mathcal{E})\varphi'_{op} - X(\mathcal{E}) = 0 \qquad \textbf{78}$$

b) Del vector obtenido φ'_{op} retirar el parámetro con el valor más pequeño en magnitud.

$$\varphi'_s = \min \left\| \varphi'_{op} \right\| \quad (79)$$

c) Volver a generar el sistema de ecuaciones retirando las filas asociadas a ese parámetro tal que:

$$\varphi'_R = \varphi'_R \backslash \varphi'_s \quad (80)$$

$$W_{\varphi_N}(\mathcal{E}) = W_{\varphi_R}(\mathcal{E}) \backslash W_{\varphi_s}(\mathcal{E}) \quad (81)$$

d) Recalcular el vector de parámetros φ'_N

$$min \left\| \varphi'_N \right\|_2 \quad sujeto\ a \quad W_{\varphi_N}(\mathcal{E})\varphi'_N - X(\mathcal{E}) = 0 \quad (82)$$

e) Recalcular el vector de fuerzas exteriores $X(\mathcal{E})_N$

$$X(\mathcal{E})_N = W_{\varphi_N}(\mathcal{E})\varphi'_N \quad (83)$$

f) Calculamos el error:

$$\varepsilon = \frac{\|X(\mathcal{E}) - X(\mathcal{E})_N\|}{\|X(\mathcal{E})\|} \quad (84)$$

g) En este punto se plantea un nuevo sistema de ecuaciones, esta vez con el sistema reducido que puede ser expresado como:

$$W_{\varphi_N}(\mathcal{E})\varphi'_N = X(\mathcal{E})_N; \quad (89)$$

h) Este sistema de ecuaciones es resuelto nuevamente bajo la norma L1 teniendo:

$$min \left\| \varphi'_N \right\|_1 \quad sujeto\ a \quad W_{\varphi_N}(\mathcal{E})\varphi'_N - X(\mathcal{E})_N = 0 \quad (90)$$

i) En este punto se retira el parámetro con el valor más pequeño en magnitud del vector

$$\varphi'_s = \min \left\| \varphi'_N \right\| \quad (91)$$

Y se vuelven a aplicar las operaciones de la c a la i hasta retirar todos los parámetros.

En este método se tienen de alguna manera casi las mismas desventajas que en el apartado anterior. Sin embargo, estas han sido disminuidas debido a que el orden en él que se retiran los parámetros es alterado. De manera que antes de quitar alguno, revisamos que tan importante es este para el modelo mediante la aplicación continua de la norma L1. Por tanto, de una manera más óptima se van retirando los parámetros del modelo dinámico. El aplicar este método de reducción de parámetros ha ofrecido mejores resultados que en algunos casos (dependiendo del error a considerar) llegan a ser mejores que los métodos que se han llevado hasta ahora.

4. Ejemplos y simulaciones:

La alta o baja movilidad de un cuerpo en un sistema multicuerpo es definido como la posibilidad del cuerpo de exhibir grandes o pequeñas excursiones en su espacio de movimiento relativo respecto de otros cuerpos y con respecto del marco de inercial. Esto juega un rol importante en la relevancia de diferentes parámetros dinámicos. Por ejemplo en [40] se muestra que para cada grado de libertad perdido entre dos cuerpos arbitrarios, o entre un cuerpo arbitrario y su marco de inercia, el número de parámetros base del sistema puede ser reducido, por tanto, si algún movimiento relativo entre dos cuerpos o entre un cuerpo y el suelo tiene una amplitud pequeña, la relevancia de algunos parámetros en la dinámica puede ser pequeña. Por lo tanto, en estos casos, se espera obtener un modelo de parámetros reducidos preciso con un número de parámetros relativamente pequeños.

De allí la motivación en proponer ensayos para diferentes estrategias de reducción de parámetros en dos diferentes sistemas multicuerpo con diferentes tipos de movilidades características (alta o baja).

4.1 Ejemplo de sistema multicuerpo con alta movilidad:

Los 6 grados de libertad del manipulador PUMA560 en serie mostrado en la Figura 1, .PUMA Por sus siglas en ingles es definido como Programmable Universal Machine for Assembly (maquina programable universal para ensamblajes). Es un brazo robótico industrial desarrollado por Víctor Scheiman. Todos los diseños consisten en dos componentes principales: el brazo mecánico y el sistema de control. Estos son típicamente interconectados por uno o dos cables multiconductores. Cuando dos de los cables son usados uno lleva la potencia a los servomotores y frenos mientras que el segundo lleva la posición y el retorno para cada junta al sistema de control.

Estos robots son usados en la industria para diferentes actividades entre ellas el ensamblaje de partes, movimiento de material y soldadura. El modelo de Sistema multicuerpo desarrollado para este robot puede ser usado no solo para su diseño, en el cual se puede conocer las cargas que sufrirán cada uno de los componentes, sino también puede ser utilizado para su control. En este caso las

estrategias de control basado en modelo requieren de este, por ejemplo, las fuerzas que habrán de ejercer los actuadores para seguir una trayectoria deseada. Este cálculo debe ser realizado miles de veces por segundo. De ahí la necesidad de modelos precisos pero suficientemente sencillos para que puedan ejecutarse a la frecuencia deseada.

Como veremos más adelante el Sistema Multicuerpo está constituido con alrededor 60 parámetros. El cálculo de las trayectorias y fuerzas resultantes de los modelos de simulación toman tiempo. Estos son hechos en base a los sensores de retornos recibidos por la máquina y las órdenes de trabajo. Por tanto, el tiempo de respuesta para su diseño y control tiene que ser muy bajo.

Es muy claro el interés de aumentar este tiempo de respuesta, por tanto tener modelos equivalentes que lleven a un aumento en este tiempo de respuesta se expresará como un posible aumento en la producción. Hay que tener en cuenta que estos modelos “equivalentes “, que son más rápidos, son menos precisos. Por tanto hay que tener un compromiso entre el modelo reducido, el aumento en la velocidad de cálculo y la precisión.

Un claro ejemplo de un sistema multicuerpo de alta movilidad es el PUMA. Los parámetros cinemáticos modificados de Denavit-Hartenberg [41], las coordenadas, y el conjunto de parámetros de inercia usados para definir el modelo, son tomados como referencia de [42], estos son mostrados en las tablas 1 y 2 respectivamente tal como en [2].

Figura 1 Ejemplo de alta movilidad: Izq.) Sistema Actual, Der.) Modelo CAD

El Principio de las Potencias Virtuales ha sido utilizado para obtener las ecuaciones dinámicas. Como coordenadas juntas/relativas son usadas para el modelado, no se requieren ecuaciones de restricción ni multiplicadores de Lagrange. La librería simbólica en [43] ha sido usada para obtener las funciones

de IDM (Inverse Dynamic Model) y DDM (Direct Dynamic Model), $\boldsymbol{d_z(z,\dot{z},\ddot{z},\varphi_R)}$ y $\boldsymbol{[M_{ZZ}|\delta_z](z,\dot{z},\varphi_R)}$, respectivamente. Esta librería minimiza fuertemente el número de operaciones algebraicas usadas en algoritmos compatibles con formulaciones recursivas de tipo $O(n^3)$, con un desempeño que está a la par con formulaciones de vanguardia. De esta manera, las simplificaciones reportadas son realizadas sobre modelos cuasi-óptimos. Esto asegura un reporte justo de la complejidad computacional de los modelos reducidos, los cuales son dados en términos de números de operaciones.

Los valores de Torques nominales para los actuadores $nom(\tau_z) = [350,300,125,8,3,1]\,Nm$, son usados para definir la predicción del error normalizada para IDM,

Tabla 1 Parámetros modificados de Denavit-Hartenberg para el PUMA560

Body Units	ai m	αi rad	di m	θi rad
1	0	0	0	z1
2	0	−π/2	0	z2
3	0.4318	0	-0.1491	z3
4	-0.0203	π/2	-0.4318	z4
5	0	−π/2	0	z5
6	0	π/2	0	z6

Tabla 2 Parámetros dinámicos para el PUMA560

Body Unit	m kg	dx kg m	dy kg m	dz kg m	Ixx kg m2	Ixy kg m2	Ixz kg m2	Iyy kg m2	Iyz kg m2	Izz kg m2
1	10.52	0.0	−0.568	0.0	1.643	0.0	0.0	0.509	0.0	1.643
2	15.78	2.206	0.2	2.353	0.841	0.2	−0.329	8.738	0.4	8.576
3	8.767	−0.003	−1.727	0.0	3.717	−0.001	0.002	0.301	0.002	3.717
4	1.052	0.03	0.06	−0.060	0.184	0.000	0.0	0.184	0.000	0.127
5	1.052	0.004	−0.007	0.005	0.074	0.000	0.000	0.074	0.000	0.127
6	0.351	0.01	0.02	0.013	0.008	0.000	0.002	0.008	0.000	0.014

4.1.1 Trayectorias de estimación:

El número condicionado de una matriz es una propiedad que representa la relación que hay entre el valor singular más grande de la matriz con respecto al valor singular más pequeño [44]. El número condicionado de la matriz de observación $k\big(W(\mathcal{E})\big)^2$ ha sido utilizado como función objetivo para la optimización de las trayectorias. Las coordenadas independientes del robot, $z_1,\ldots,z_6$, han sido parametrizadas usando una serie finita de Fourier, como se propone en [45] . Un total de 100 puntos de muestras iguales cubriendo un periodo completo son extraídos de cada trayectoria optimizada. Para obtener un conjunto de muestras características que exciten suficientemente el sistema, sobre unas condiciones de optimización definida, los datos muestreados de 10

diferentes trayectorias de optimización son usados para la optimización. La función *fmincon* de MATLAB Optimization Toolbox ha sido empleada para realizar la optimización actual. Los ángulos de rotación y los rangos de velocidad angular de los actuadores han sido limitados usando restricciones lineales de desigualdades. Por referencia, los detalles de las trayectorias de optimización son resumidas en la tabla 3. Para dar una idea gráfica, una de las trayectorias optimizadas es mostrada en la figura 2.

Tabla 3 EAM: definición de trayectorias de excitación.

Criterio de optimización	Numero de cond.
# Juntas actuadoras	6
# Armónicos	4
# Parámetros de trayectoria	54
# Puntos de muestra por trayectoria.	100
# Restricciones de desigualdades lineales	2400
# Restricciones de desigualdades no-lineales	0
Periodo de trayectoria principal	2π s
Z_{min}	−π/2 rad
Z_{max}	π/2 rad
$\dot{Z}_{min}$	−1.45 rad/s
$\dot{Z}_{max}$	1.45 rad/s
# Trayectorias estimada	10
# Trayectorias validadas	1

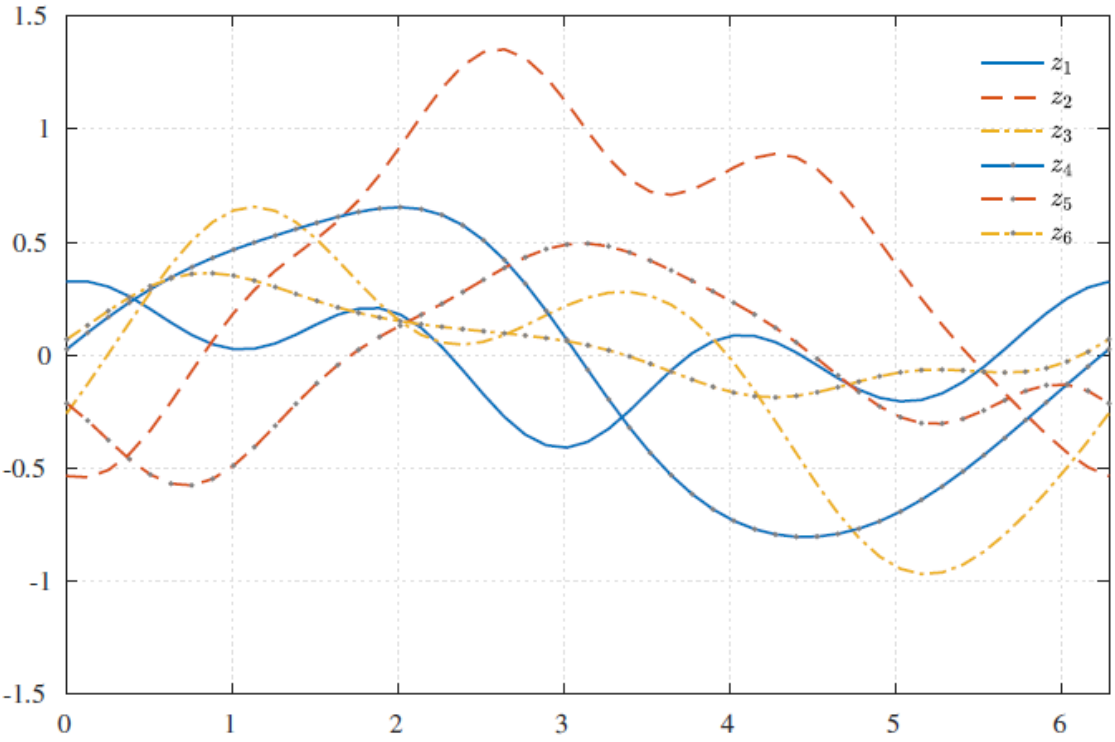

Figura 2 Ejemplo de la trayectoria caracteristica $(\mathcal{V})$. El tiempo en (s) es mostrado en el eje X

4.1.2 Desempeño de los modelos de reducción de parámetros:

Para medir el desempeño obtenido con los algoritmos propuestos en este trabajo, empezaremos con graficar el tanto el desempeño de los algoritmos propuestos en [2] como los algoritmos desarrollados en este trabajo, usando en cada uno en el mismo contexto en este caso en el PUMA560. (Sistema Multicuerpo de alta movilidad).

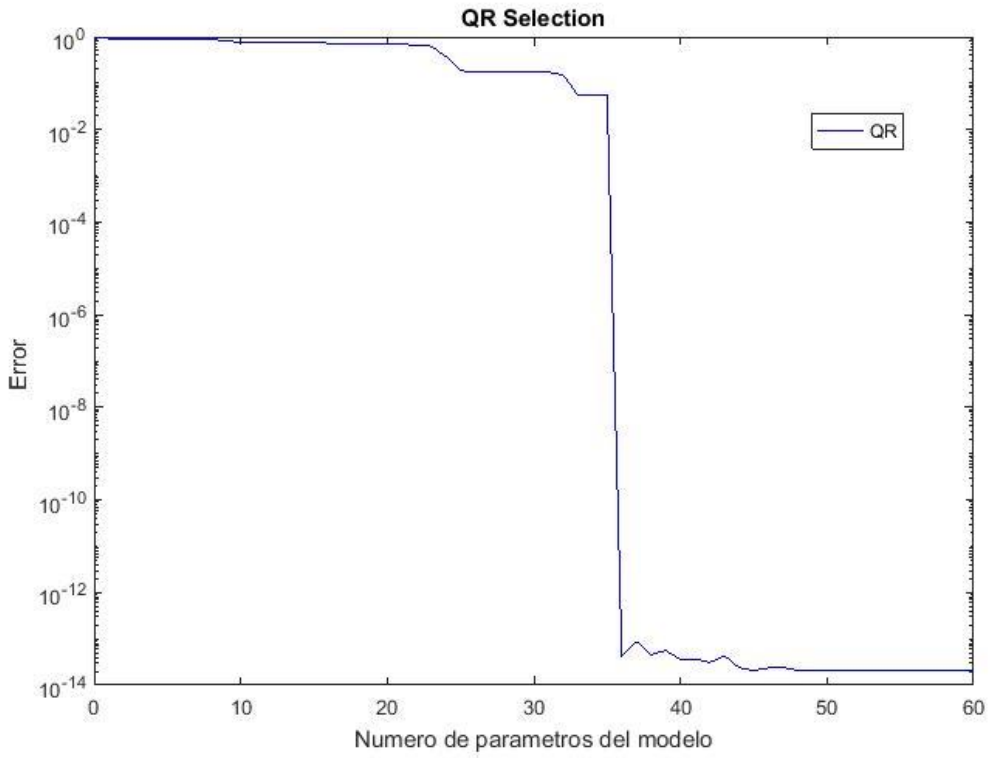

Figura 3 PUMA560: QR selection Error, vs nº de parámetros

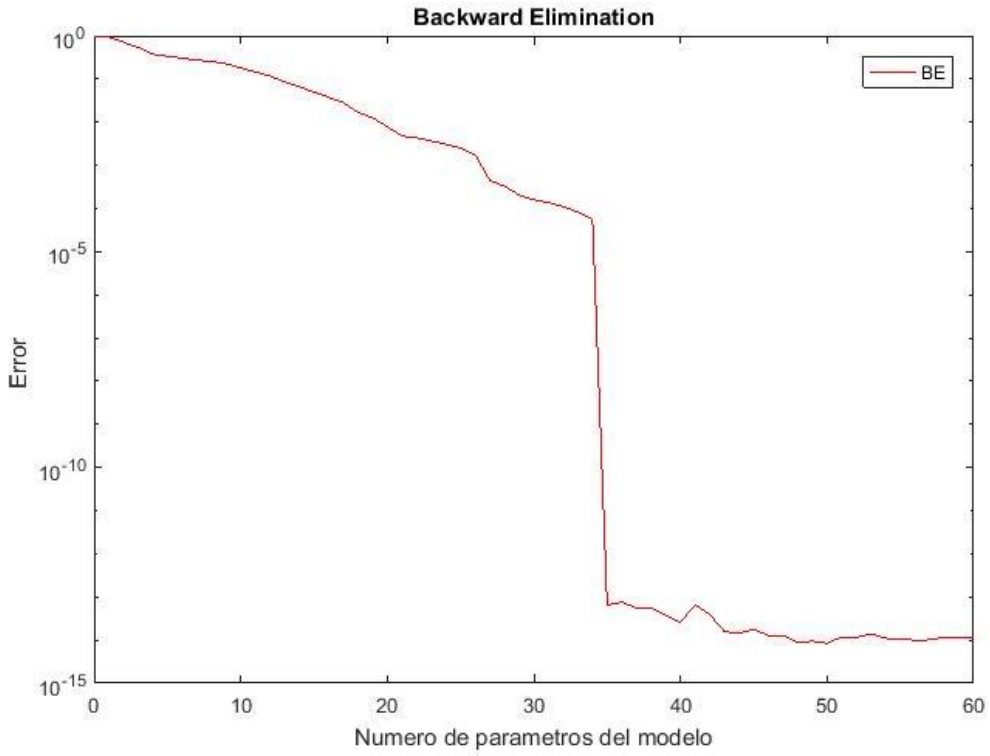

Figura 4 PUMA560: Backward Elimination, Error vs nº de parámetros

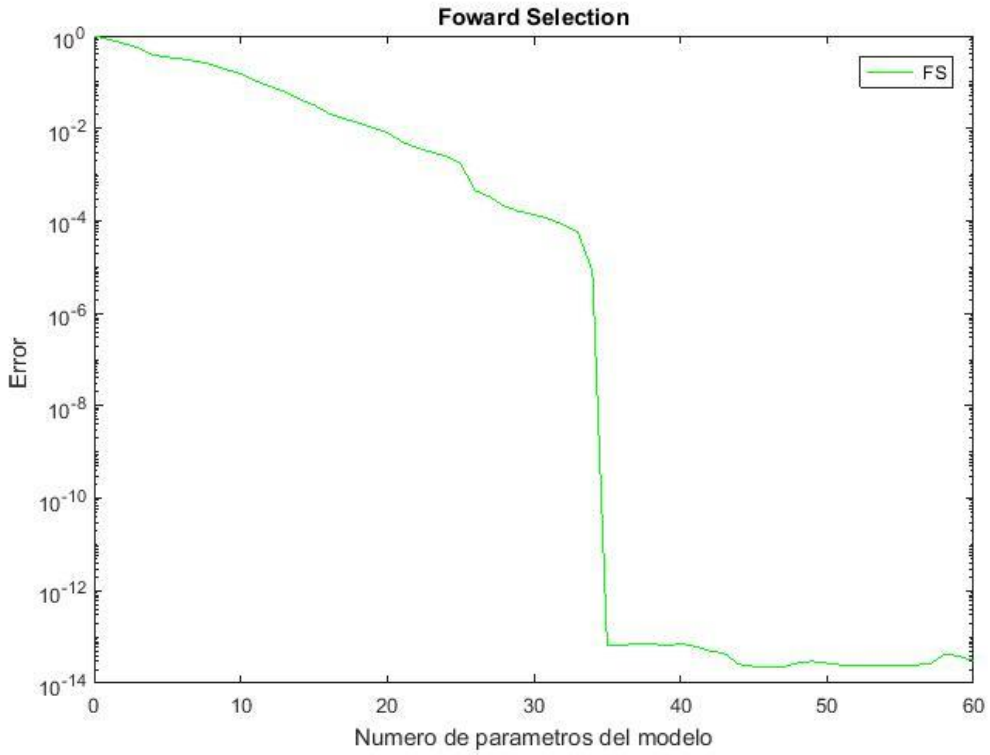

Figura 5 PUMA560: Foward Selection, Error vs nº de parámetros

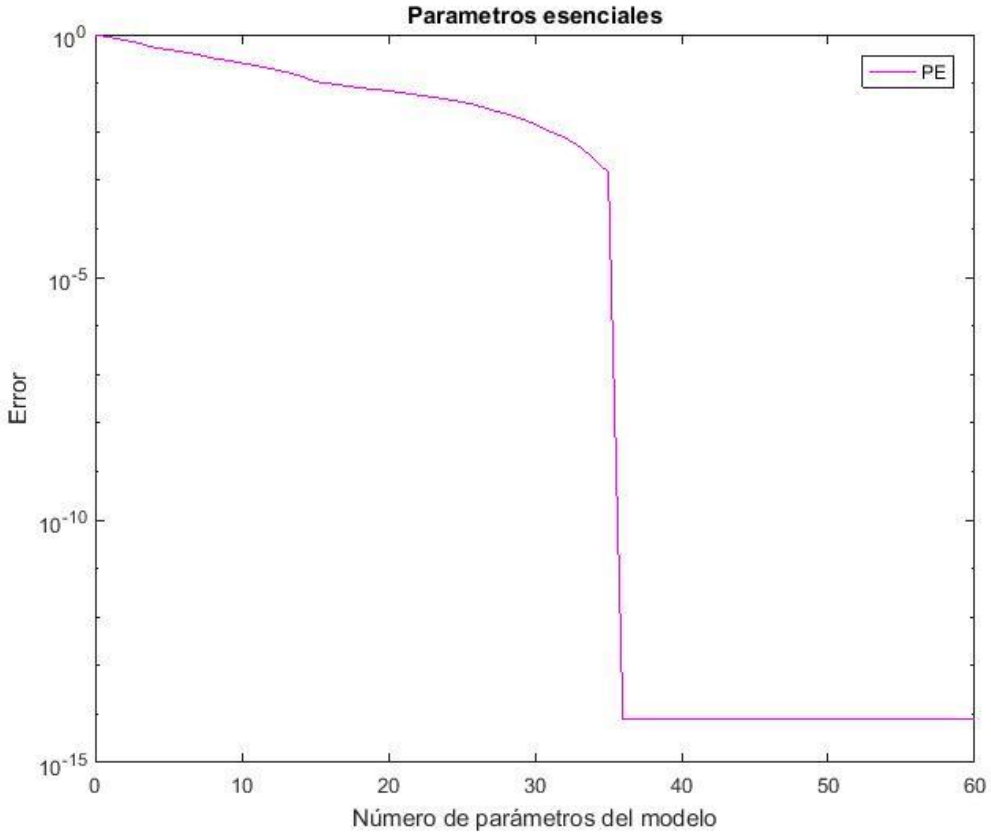

Figura 6 PUMA560: Parámetros Esenciales, Error vs nº de parámetros

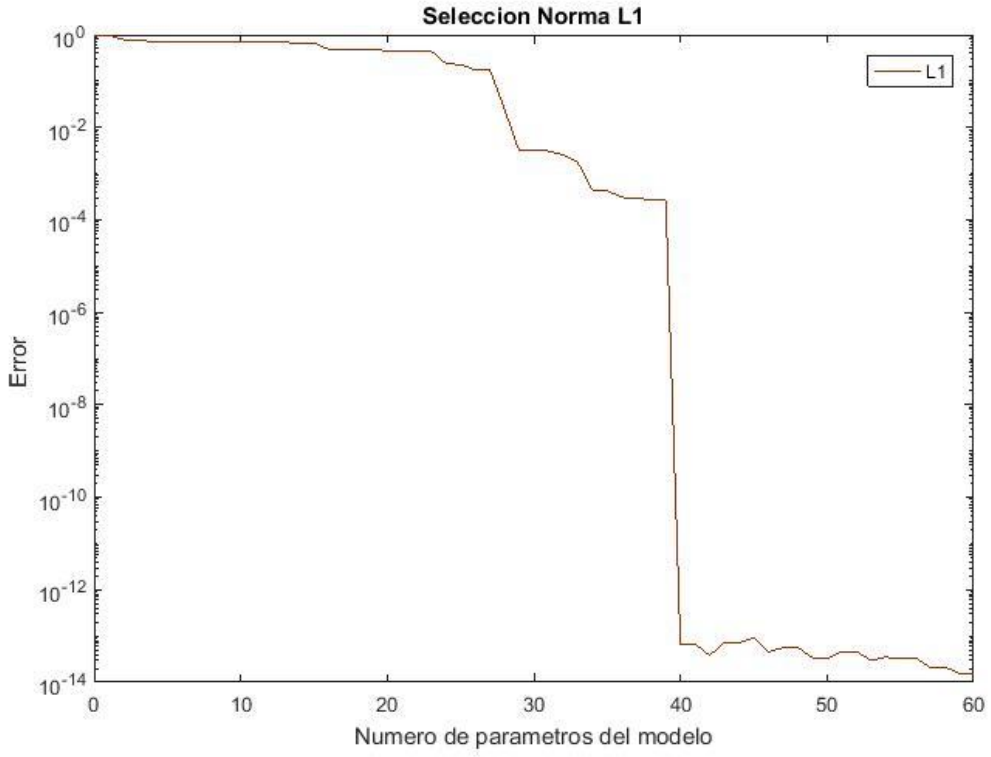

Figura 7 PUMA560: Selección Norma L1, Error vs nº de parámetros

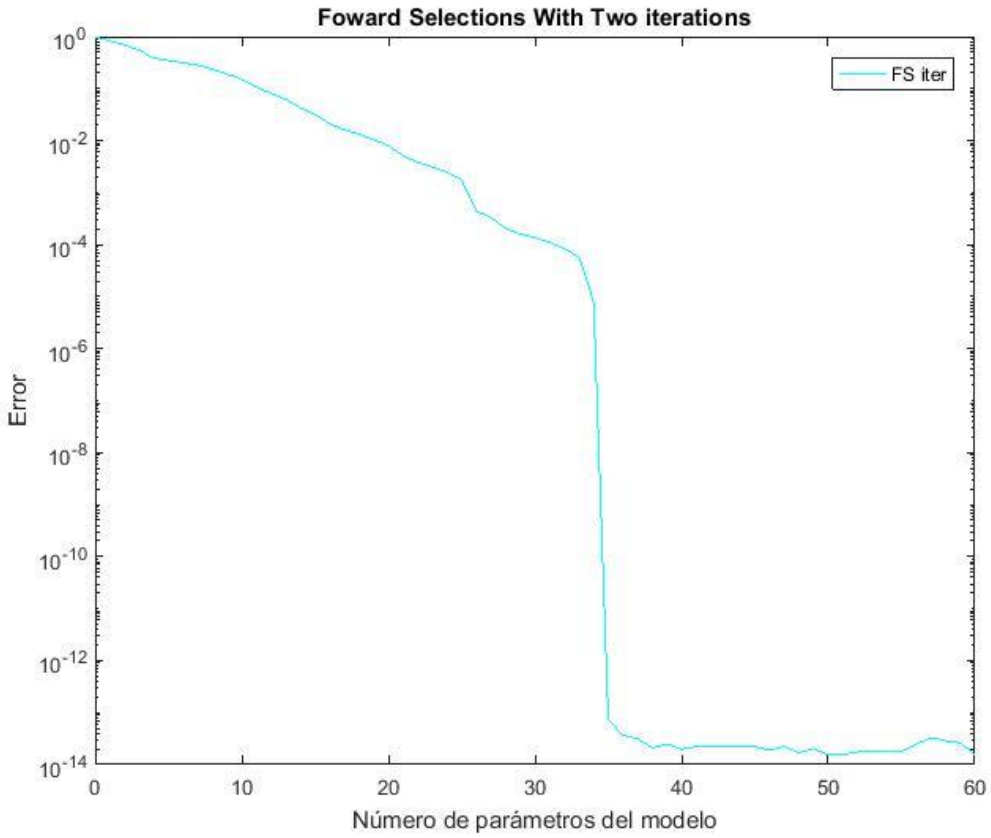

Figura 8 PUMA560: Foward Selection Iterativo, Error vs nº de parámetros

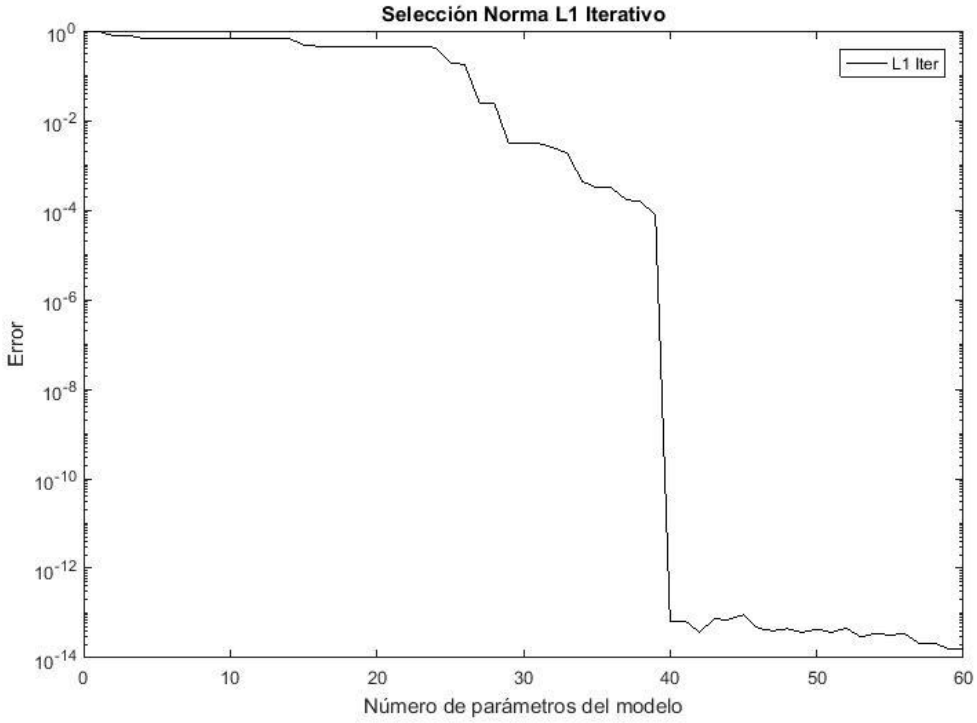

Figura 9 PUMA560: Selección Norma L1 Iterativo, Error vs nº de parámetros

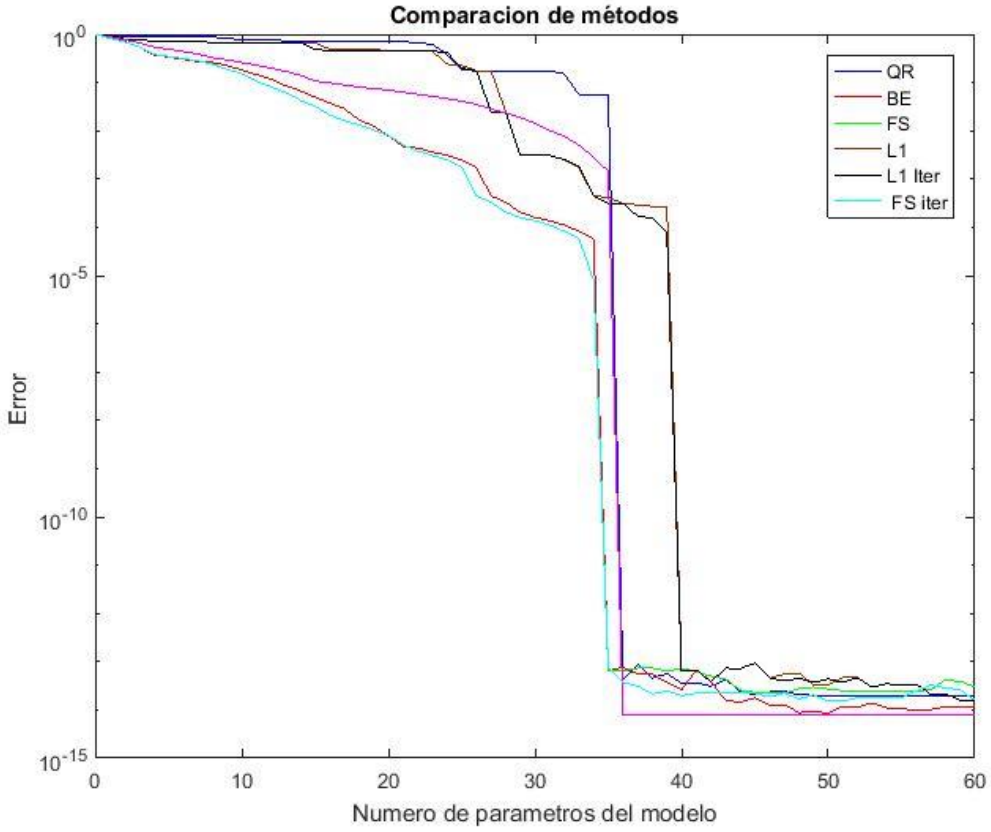

Figura 10 PUMA560: Comparativa de métodos, Error vs nº de parámetros

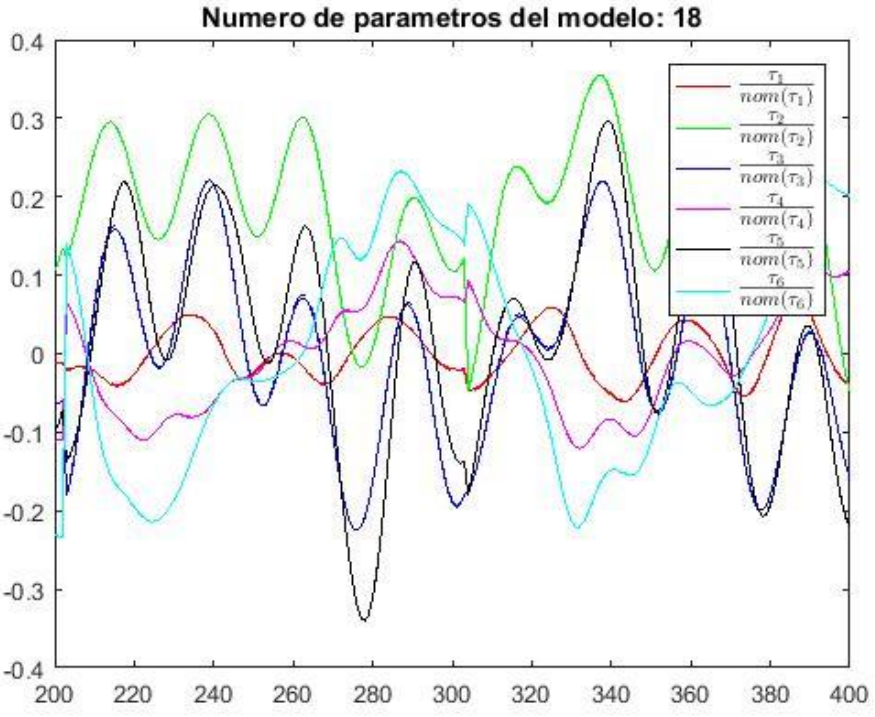

Figura 11 PUMA560: Modelo seleccionado (FS $n_{\varphi R} = 18$)$\frac{\tau_i(\varphi_R)}{nom(\tau_i)}$(- -) y $\frac{\tau_i(\varphi)}{nom(\tau_i)}$(--) vs $T(S)$

Tabla 4 PUMA560: Reordenación de parámetros por cada método.

Orden de importancia al modelo	Norma L1 iterativa	Norma L1	FS	BE	QR
1	Iyy2	Izz2	m3	my3	my6
2	Izz2	Iyy2	Izz5	m5	mx6
3	mx2	mx2	Izz3	Izz5	Ixy6
4	Ixx4	Izz1	my6	my6	Iyz6
5	Iyy4	Izz3	mx6	mx6	Ixz6
6	Ixx3	Ixx3	mz6	mz5	Izz6
7	Izz3	Iyy4	mx2	mz6	my5
8	Izz1	Ixx4	my4	Iyy5	mx5
9	my3	my3	Izz4	Ixx5	Iyy6
10	Iyz2	Iyz2	my3	Izz2	Ixx6
11	Ixz2	Ixz2	Izz6	Izz6	Iyz5
12	Ixy2	my2	Ixx3	Iyy2	Ixz5
13	my2	Ixy2	Ixz2	mz4	my4
14	Ixx6	Iyy5	mx4	mx4	Ixy5
15	Iyy5	Ixx5	Iyy4	mx3	mx4
16	Iyy6	Ixx6	mx5	Ixx4	Ixx5
17	my4	Iyy6	mz3	mx5	Iyz4
18	Ixx5	Izz5	my2	Ixx2	Ixz4
19	Izz5	mx3	m4	Ixz2	Iyy5
20	Izz4	Izz4	Ixx5	my2	Ixy4
21	mx3	mz5	Iyz2	Iyz2	Iyy4
22	mx4	my4	Ixx2	Ixy3	my3
23	mz5	mx4	Ixy2	Ixy2	mx3
24	Ixy3	Ixy3	Iyz6	Izz3	mx2
25	mz6	my6	Ixy6	Iyz6	Izz4
26	my6	Izz6	Ixz6	Ixy6	my2
27	Izz6	my5	Ixz5	Ixz6	Ixz3
28	mx6	mz6	Iyz5	Ixz5	Iyz3
29	my5	mx6	Ixy5	Iyz5	Ixy3
30	mx5	mx5	Iyz4	Ixy5	Ixz2
31	Ixz3	Iyz3	Ixz4	Iyz4	Iyz2
32	Iyz3	Ixz3	Ixz3	Ixz4	Izz1
33	Ixy6	Iyz6	Iyz3	Ixz3	Izz2
34	Ixz6	Ixy6	Ixy4	Iyz3	Ixy2
35	Iyz6	Ixz6	mx3	Ixy4	Izz3
36	Ixz5	Ixy5	m1	m2	Ixx2
37	Ixy4	Iyz5	mx1	Iyz1	Izz5
38	Ixz4	Iyz4	my1	Iyy1	mz6
39	Ixy5	Ixy4	Iyy6	Ixz1	Iyy3
40	Iyz4	Ixz4	Ixy3	Ixy1	mz5

Orden de importancia al modelo	Norma L1 iterativa	Norma L1	FS	BE	QR
41	Iyz5	Ixz5	Ixx4	Ixx1	mz4
42	m3	m3	Iyy3	mz1	Ixx4
43	m6	m6	Izz2	my1	m5
44	m5	m5	Iyy2	mx1	m3
45	m4	m4	m5	Iyy6	Iyy2
46	mz4	mz4	mz4	Ixx3	mz3
47	mz3	mz3	mz1	m1	Ixx3
48	Ixx2	m2	Ixx6	mz2	m4
49	Iyy3	m1	Iyy5	my5	m6
50	mz2	mz2	Izz1	Izz1	Ixz1
51	m2	Ixx1	Ixx1	mz3	m2
52	Iyz1	Ixx2	Ixy1	m6	mx1
53	Iyy1	Iyy3	Ixz1	mx2	m1
54	Ixz1	my1	Iyy1	Ixx6	mz2
55	Ixy1	Iyy1	Iyz1	m3	Iyy1
56	Ixx1	Iyz1	m2	my4	my1
57	mz1	Ixz1	mz2	Iyy3	Ixx1
58	my1	Ixy1	mz5	Iyy4	Iyz1
59	mx1	mz1	my5	Izz4	mz1
60	m1	mx1	m6	m4	Ixy1

Como es de notar en la tabla 4, los parámetros escogidos por el método de los parámetros esenciales y el método de FS iterativo no han sido obtenidos, debido a que no pueden ser usados para simplificar el modelo dinámico real de la misma manera que en los otros métodos. En este caso se tienen diferentes soluciones para un determinado número de parámetros, que pueden contener parámetros de otras soluciones. En el caso de los parámetros esenciales sus cálculos han sido sólo hechos para dar una idea de que tanto puede reducirse un modelo y establecer una línea de cota, sin embargo ha sido notable el hecho de que para este caso, muchos de los métodos aquí presentados, son mejores que este método que es, considerado estándar de facto en la reducción de modelos de la robótica industrial.

4.2 Ejemplo de sistema multicuerpo con baja movilidad

Las dificultades particulares del mecanismo de identificación paralela son muy conocidas. Por ejemplo en [46] los problemas asociados con la suspensión de un automóvil son analizados. Estos son frecuentemente relacionados con la presencia de lazos cerrados que limitan la movilidad de algunos cuerpos a una extensión significativa. En este estudio, el 6-PUS Hexaglide, mostrado en la

figura 11 es el seleccionado como el ejemplo de un sistema multicuerpo de baja movilidad (SMBM).

El Hexaglide, es definido como un robot paralelo, que se compone de dos plataformas interconectadas, en donde se provee movimiento relativo de una plataforma movible con respecto, a una plataforma base. Su uso en la industria, se encuentra más que todo en el fresado de alta velocidad de elementos, y piezas mecánicas. Debido a la alta velocidad requerida en estas maquinarias el control, y el diseño de los elementos que transmiten el movimiento, requieren modelos o simulaciones que puedan ser lo suficientemente rápidas.

Para los modelos de sistemas multicuerpos, que simulan estos componentes, ha de ser importante, el acelerar los cálculos de fuerzas y trayectorias, ya que esto se transmite en un aumento de la productividad en la industria. En este caso en particular el robot se compone de 70 parámetros. El reducir estos modelos mediante estos métodos, lleva igualmente que en el caso del PUMA, un compromiso entre, el número de parámetros y la precisión.

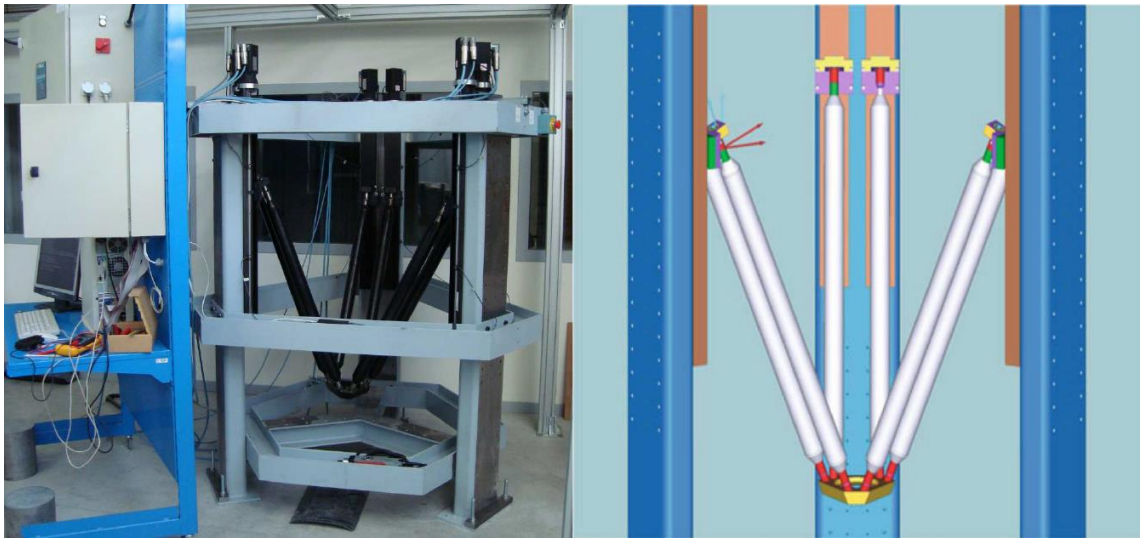

Figura 12 Hexaglide: Izq.) Prototipo, Der.) Modelo CAD

Un total de 24 coordenadas generalizadas, **q**, son usadas para modelar el sistema: las alturas de las 6 guías, $\boldsymbol{z} = [z_1, \ldots, z_6]$, son las coordenadas independientes, 2 rotaciones relativas de las juntas universales para cada uno de los 6 brazos, y 6 coordenadas absolutas para la cabeza (las coordenadas cartesianas del centro de la cabeza, y sus ángulos de Euler). Para propósitos de referencia, los parámetros cinemáticos del prototipo son mostrados en la tabla 5.

Tabla 5 Parámetros cinemáticos del Hexaglide

Longitud de barra	L	1.0000 m
Separación de guía lineal	e	0.1365 m
Separación de junta de cabezal	e	0.1365 m
Eje de simetría de la estructura a la junta U	R	0.4840 m
Eje de simetría de la cabeza a la junta S	r	0.0730 m
Simetría equilateral	α	2π/3 rad

La naturaleza del sistema de lazo cerrado requiere forzar 18 restricciones (3 para cada junta en el cabezal), llevando a un total de 6 grados de libertad. Igual que en el caso anterior, el principio de las potencias virtuales ha sido usado para obtener las ecuaciones dinámicas. Los parámetros de inercia de las barras, referenciadas a las juntas U con los carros, y los de la cabeza, referenciada al centro de las juntas S, son dadas en la tabla 6.

Tabla 6 Parámetros dinámicos del Hexaglide

Body	m	dx	dy	dz	Ixx	Ixy	Iyy	Ixz	Izz	Iyz
Unit	kg	kg m	kg m	kg m	kg m2	kg m2	kg m2	kg m2	kg m2	kg m2
Head	6.697	0.07	0.07	−0.238	0.0283	0.001	0.028	0.000	0.038	0.000
Bar	5.804	0.03	0.03	−1.469	1.044	0.000	1.044	0.014	0.002	0.014

Las restricciones requeridas son realizadas usando multiplicadores de Lagrange $\boldsymbol{\lambda}$**,** llevando al siguiente conjunto de ecuaciones dinámicas:

$$d_q = M_{qq}\ddot{q} - \delta_q + \phi_q^T\lambda = \tau_q \quad \textbf{92}$$

Donde $\phi_q = \frac{\partial\phi}{\partial q}$ es el jacobiano de las ecuaciones de restricción geometricas. Esta ecuación puede ser reordenada como:

$$d_q = K_{q\varphi}\varphi + \phi_q^T\lambda = \tau_q \quad \textbf{93}$$

El vector de coordenadas generalizadas es ordenado como $q = [d^T, z^T]^T$**,** Donde **z** es el conjunto de coordenadas independientes (coordenadas de los actuadores lineales). En consecuencia, las columnas jacobianas son ordenadas como $\phi_q = [\phi_d, \phi_z]$.

Ahora notar que:

$$R = \begin{bmatrix} -\phi_d^{-1}\phi_z \\ 1 \end{bmatrix} \quad \textbf{94}$$

Es un complemento ortogonal de $\phi_q, \phi_q\ R = 0$**,** multiplicando, ec. **89** y ec.**88** Por R^T y sustituyendo tenemos:

$$\ddot{q} = R\ddot{z} + \begin{bmatrix} \phi_d^{-1}\gamma \\ 0 \end{bmatrix} \quad \textbf{95}$$

Donde $\gamma = \phi_q\ddot{q} - \ddot{\phi}$, en la ec **88**, estas ecuaciones toman la forma de las ecs. **3** y **5**. Llevando a la formulación en coordenadas independientes de las funciones dinámicas escogidas para describir los algoritmos.

$$d_z = R^T d_q \quad \textbf{96}$$

$$M_{zz} = R^T M_{qq} R \quad \textbf{97}$$

$$\delta_z = R^T \left(\delta_q - \begin{bmatrix} \phi_d^{-1}\gamma \\ 0 \end{bmatrix} \right) \quad \textbf{98}$$

$$K_{z\varphi} = R^T K_{q\varphi} \quad \textbf{99}$$

$$\tau_z = R^T \tau_q \quad \textbf{100}$$

El estado,$(q, \dot{q})$ es determinado como una función de coordenadas independientes y velocidades $(z, \dot{z})$**,** haciendo uso de las ecuaciones de restricción al nivel de posición y velocidad. Como en los ejemplos previos la librería simbólica en [43] ha sido usada para obtener las ecuaciones.

Como el procedimiento de reducción de parámetros no afecta el modelo cinemático y por lo tanto el cálculo de **R**, en [2] propone mostrar la complejidad computacional de los modelos de parámetros reducidos de este ejemplo, en términos del número de operaciones requeridas para computar $d_q(q, \dot{q}, \ddot{q}, \varphi_R)$ y $\left[M_{qq} \middle| \delta_q\right](q, \dot{q}, \ddot{q}, \varphi_R)$. Estas son matrices directamente exportadas por la librería simbólica.

Las fuerzas nominales de los actuadores $nom(\tau_z) = 1.7 \frac{2\pi}{10^{-2}} [1,1,1,1,1,1]\ N$ son usadas para definir el error de predicción normalizado ϵ_τ,

Las trayectorias son obtenidas utilizando el mismo procedimiento que en el ejemplo anterior. Las 6 coordenadas independientes $z_1, \ldots, z_6$ han sido parametrizadas usando series finitas de Fourier. En este caso el conjunto de muestras características usa datos de 25 diferentes trayectorias optimizadas. Para cada trayectoria 400 puntos de muestras son extraídos. Para propósitos de referencia, los detalles de la disposición de optimización de trayectoria fueron resumidos en la siguiente tabla:

Tabla 7 Parámetros dinámicos del Hexaglide

Criterio de optimización	Numero de cond.
# Juntas actuantes	6
# Armónicos	2
# Parámetros de trayectoria	30
# Puntos de muestra por trayectoria	400
# Restricciones de desigualdades lineales	2400
# Restricciones de desigualdades no-lineales	700
Periodo de trayectoria principal	2π s
Z_{min}	1 m
Z_{max}	2 m
$\dot{Z}_{min}$	−1.00 m/s
$\dot{Z}_{max}$	1.00 m/s
# Trayectorias estimadas	25
# Trayectorias Validadas	1

Para dar una idea gráfica, una de las trayectorias optimizadas es mostrada en la siguiente figura:

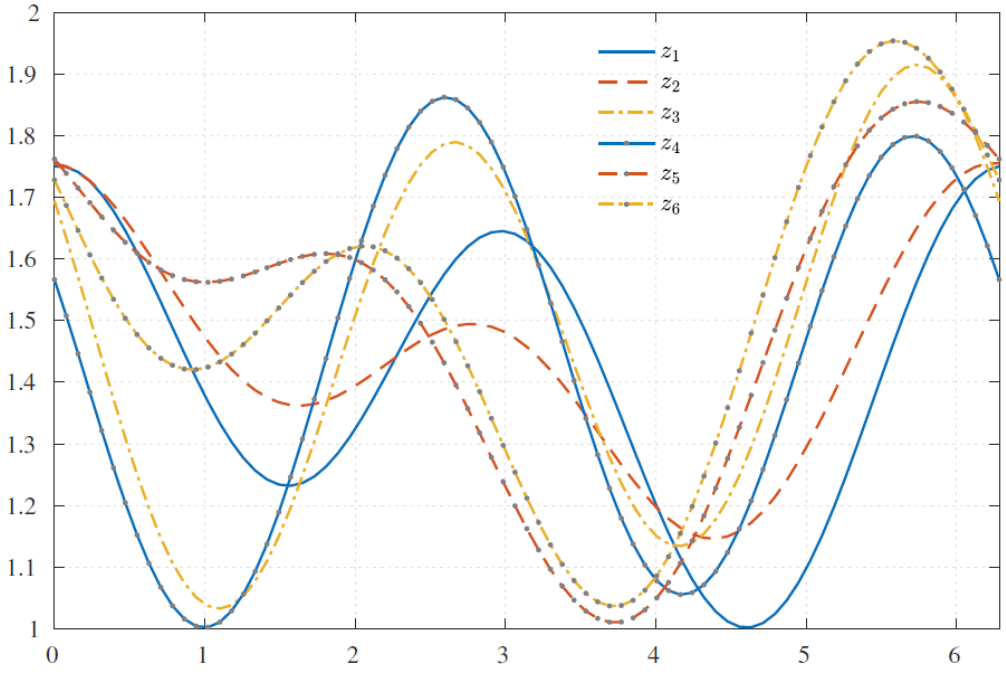

Figura 13 Ejemplo de la trayectoria caracteristica $(\mathcal{V})$. El tiempo en (s) es mostrado en el eje X

4.2.1 Desempeño de los modelos de reducción de parámetros:

Tal y como se hizo en su apartado homólogo (PUMA560) en este apartado se grafica el desempeño de, tanto los algoritmos propuestos en [2], como los algoritmos desarrollados en este trabajo, cada uno siendo aplicado en el mismo contexto en este caso en el Hexaglide. (Sistema Multicuerpo de Baja movilidad).

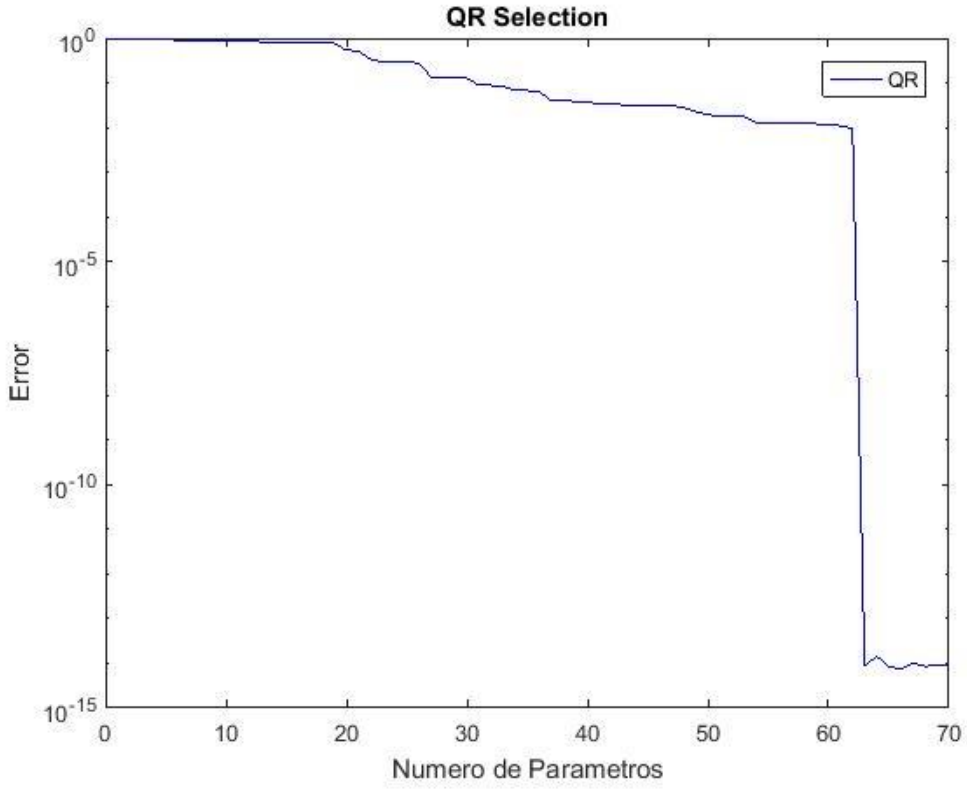

Figura 14 Hexaglide: QR selection Error, vs nº de parámetros

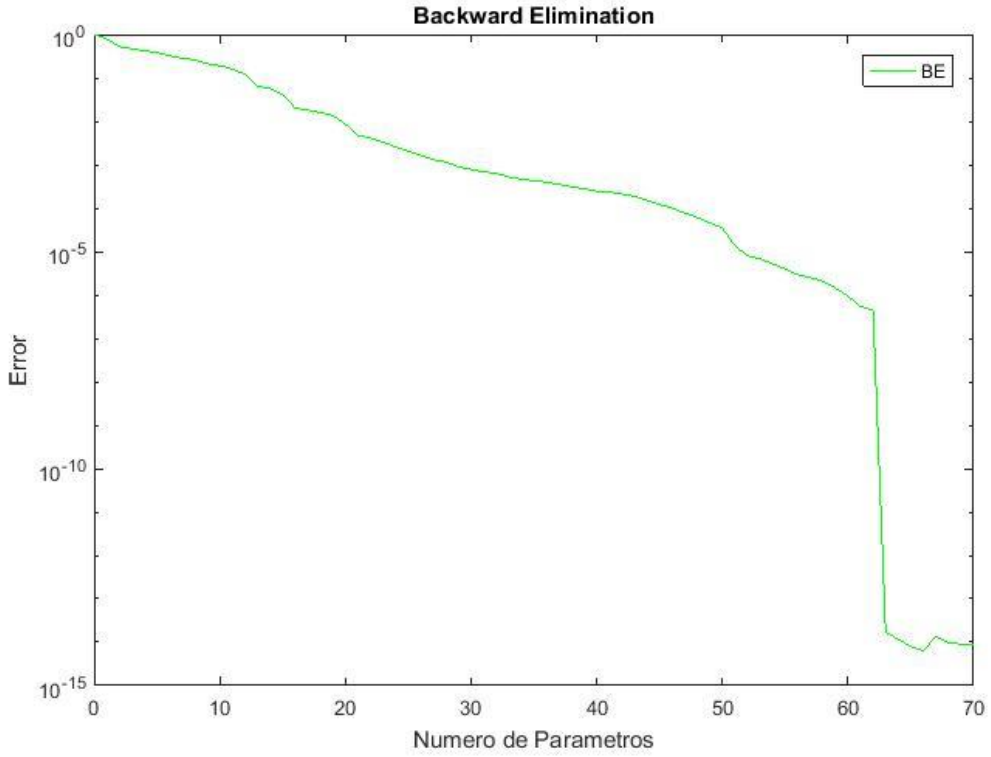

Figura 15 Hexaglide: Backward Elimination, Error, vs nº de parámetros

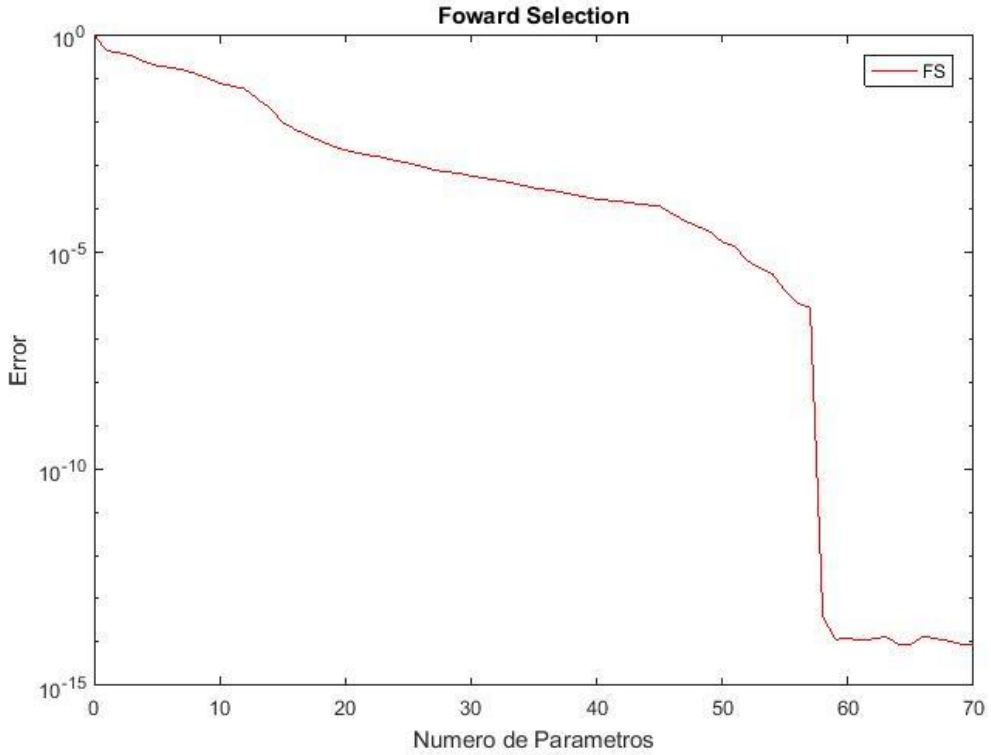

Figura 16 Hexaglide: Foward Selection, Error, vs nº de parámetros

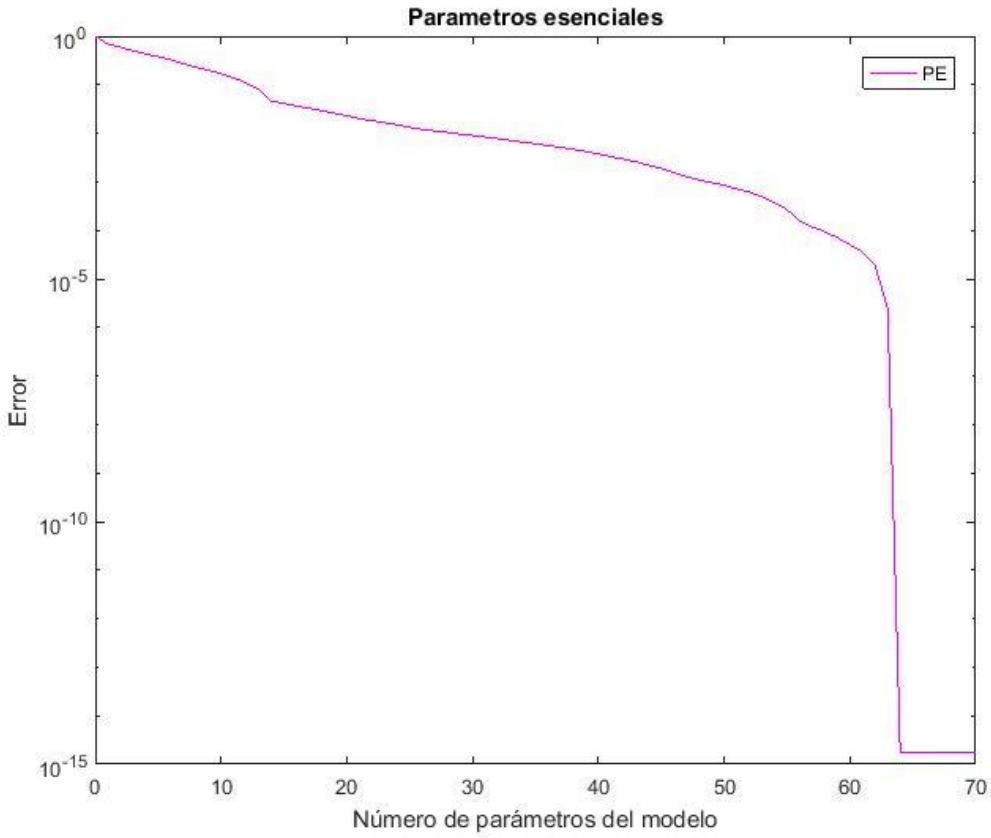

Figura 17 Hexaglide: Parámetros Esenciales, Error, vs nº de parámetros.

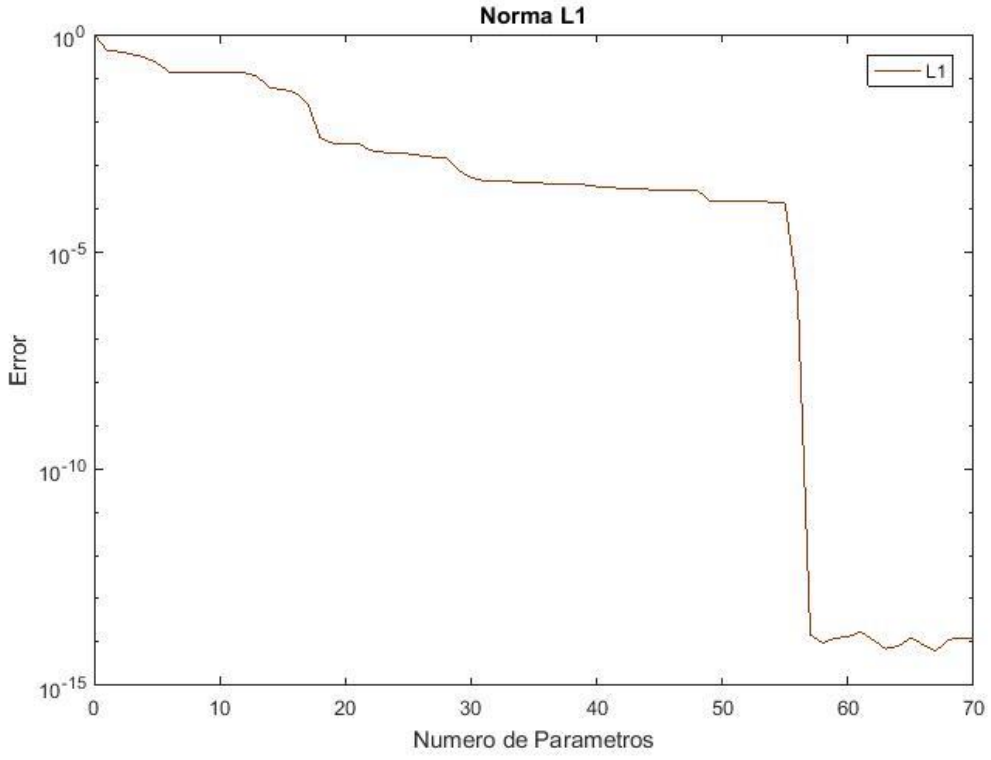

Figura 18 Hexaglide: Norma L1, Error, vs nº de parámetros.

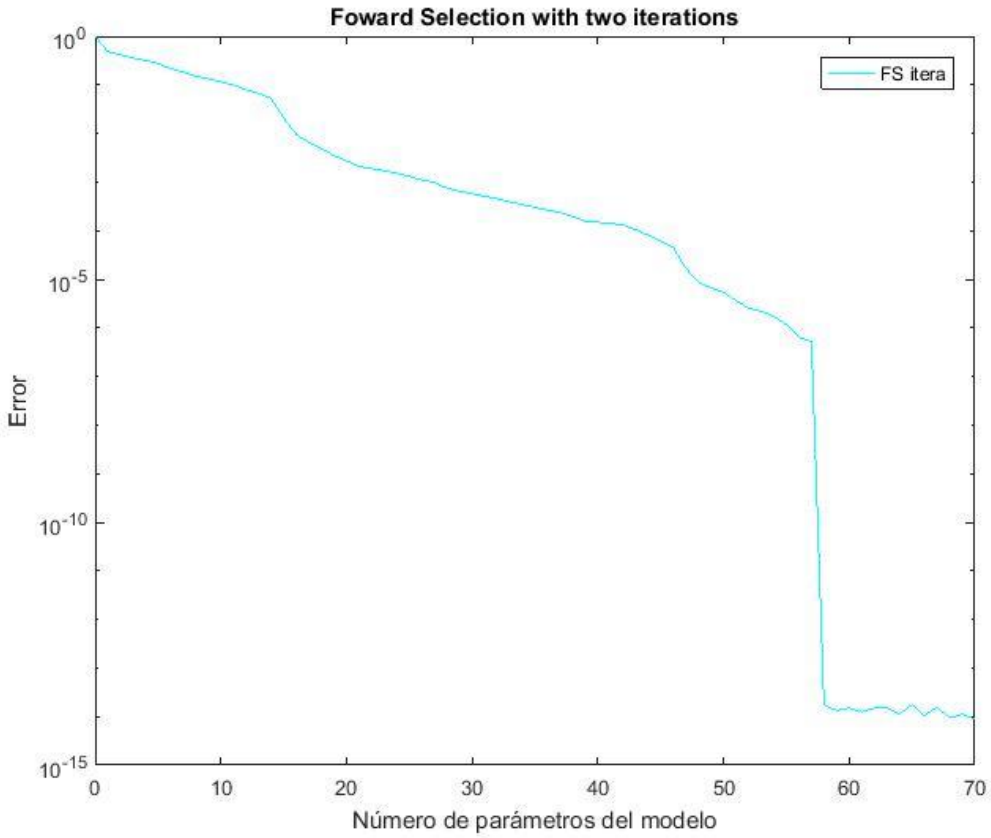

Figura 19 Hexaglide: FS iterativo , Error, vs nº de parámetros.

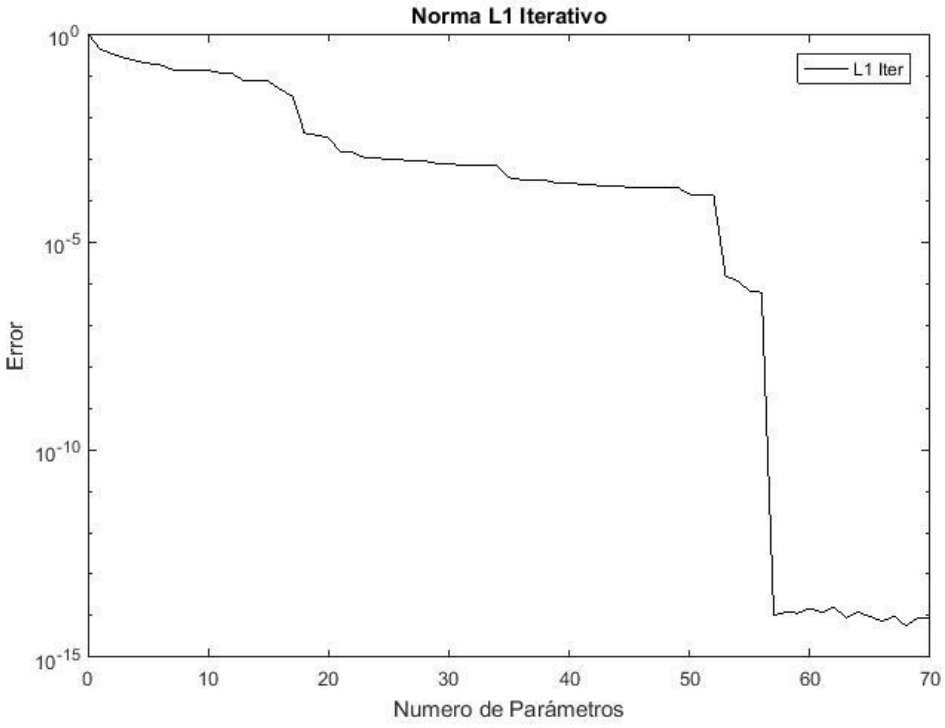

Figura 20 Hexaglide: Norma L1 iterativo, Error, vs nº de parámetros.

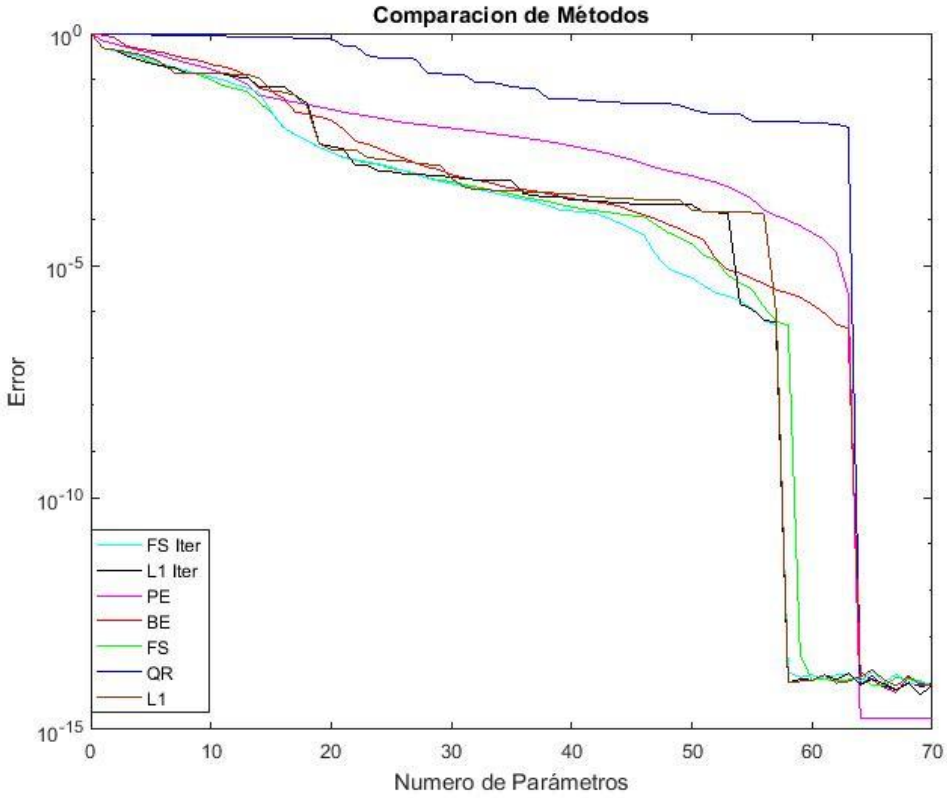

Figura 21 Hexaglide: Comparativa de metodos , Error vs nº de parámetros

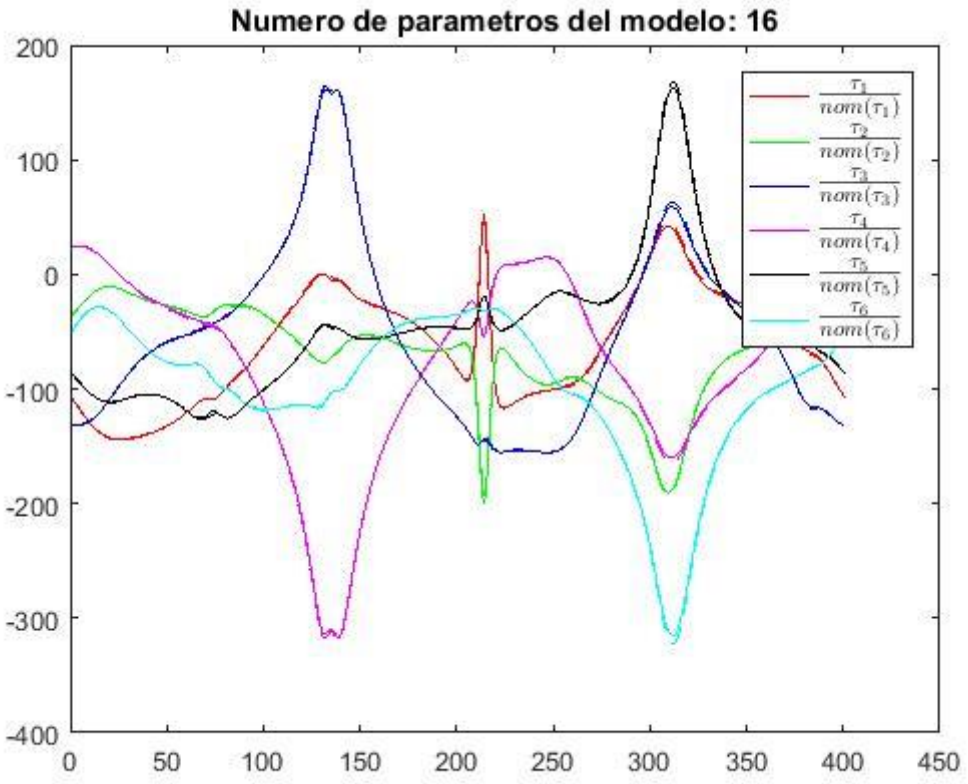

Figura 22 Hexaglide: Modelo seleccionado (FS $n_{\varphi R} = 16$)$\frac{\tau_i(\varphi_R)}{nom(\tau_i)}$(- -) y $\frac{\tau_i(\varphi)}{nom(\tau_i)}$(--) vs $T(S)$

Tabla 8 Hexaglide: Reordenación de parámetros por cada método.

Orden de importancia al modelo	Norma L1 iterativa	Norma L1	FS	BE	QR
1	mP	mP	mP	m5	IxzP
2	m4	m4	IzzP	mz5	IxyP
3	mz1	m1	mz2	myP	IxxP
4	m5	m2	m5	m6	IyzP
5	m6	m5	m4	Iyy5	mxP
6	m2	m6	mz1	mz1	mzP
7	mz4	m3	mzP	m4	myP
8	mz3	mz4	m6	Ixx1	IzzP
9	m3	mz1	mz3	mzP	IyyP
10	m1	mz2	m1	Ixx5	my3
11	mz5	mz5	my1	m2	my1
12	myP	mz6	mz6	mz3	my5
13	mxP	mz3	m2	mz6	my4
14	mzP	mzP	IxxP	Iyy1	my2
15	mz6	IzzP	IyyP	IxyP	mx3
16	mz2	mxP	m3	IxxP	mx1
17	IzzP	myP	myP	IyyP	mz1
18	IyyP	IxxP	mxP	Iyy3	mz5
19	IxxP	IyyP	mx6	Ixx3	mx4
20	my5	mx6	Ixy4	Ixx6	mz3
21	my3	mx5	my2	Iyy6	m2
22	my1	my2	mz5	mP	my6
23	Ixz3	mx4	mz4	my1	m4
24	mx4	my5	Iyy6	my3	mx6
25	Izz2	mx3	my5	my5	Ixz4
26	mx5	my6	mx4	Ixx4	mx5
27	Iyz3	my1	my3	Iyy4	mx2
28	my6	mx1	mx5	mx4	m6
29	Ixy2	mx2	Ixy3	Ixx2	Ixz1
30	my2	my4	mx2	Iyy2	Iyz4
31	Ixz2	my3	my4	mx2	Ixz2
32	mx1	Ixz4	mx1	my4	mz4
33	mx2	Iyz1	my6	mx5	Iyz1
34	Ixy5	Ixz6	Ixz4	my2	Ixz5
35	mx3	Iyz2	Ixx6	mx1	mz6
36	my4	Ixz3	Izz6	Ixz4	Ixy1
37	mx6	Iyz3	IxzP	mx6	Ixy3
38	Iyz2	Iyz5	Ixz2	my6	mz2
39	Izz3	Iyz4	mx3	mx3	Izz4
40	Izz4	Ixz1	IyzP	Ixz5	Ixy5
41	Izz6	Ixz5	Iyz3	Iyz5	Ixz3

Orden de importancia al modelo	Norma L1 iterativa	Norma L1	FS	BE	QR
42	Iyz1	Ixz2	Ixz5	Ixz1	Ixy4
43	Izz1	Iyz6	Iyz5	Iyz4	Ixz6
44	Iyz4	Izz5	Ixy1	Iyz1	Ixy2
45	Iyz6	Izz1	Ixz3	IxzP	Iyy1
46	Ixz5	Izz2	Izz1	IyzP	Ixy6
47	Iyz5	Izz4	Iyz1	Iyz3	Iyz2
48	Izz5	Izz3	Iyz4	Ixz2	Iyz5
49	Ixz4	Izz6	Iyz2	Ixz3	Ixx2
50	Ixz6	IxyP	Ixz1	Iyz2	Ixx5
51	IxyP	Ixy4	Iyz6	Ixz6	Iyy6
52	Ixz1	Ixy5	Izz4	Iyz6	Iyy3
53	IyzP	Ixy6	Ixz6	Izz4	Iyz3
54	IxzP	Ixy3	Izz2	Izz5	Izz1
55	Ixy4	Ixy2	Izz3	Izz1	m1
56	Ixy1	IyzP	Izz5	Izz3	Ixx4
57	Ixy6	IxzP	Ixy6	Izz2	Iyz6
58	Ixy3	Ixy1	Ixy2	Ixy5	Izz3
59	Iyy6	Ixx6	Ixy5	Ixy2	Izz5
60	Ixx6	Iyy6	IxyP	Izz6	Iyy5
61	Iyy5	Iyy5	Ixx1	Ixy4	Izz2
62	Ixx5	Iyy3	Ixx3	Ixy1	Ixx3
63	Iyy2	Ixx5	Ixx5	Ixy3	Izz6
64	Ixx2	Ixx3	Iyy4	Ixy6	mP
65	Ixx4	Iyy2	Iyy3	mz2	Iyy2
66	Iyy4	Ixx2	Iyy1	mxP	Iyy4
67	Ixx1	Ixx4	Iyy2	m3	Ixx6
68	Iyy1	Ixx1	Ixx2	mz4	Ixx1
69	Iyy3	Iyy1	Ixx4	IzzP	m3
70	Ixx3	Iyy4	Iyy5	m1	m5

De manera similar que en el apartado anterior. Los métodos de los parámetros esenciales y FS iterativo, no tienen el orden de importancia de los parámetros a reducir o quitar del modelo. Debido a que en estos casos la solución obtenida para un número de parámetros N, puede no estar contenida en la solución para un número de parámetros N+1.

5. Análisis de resultados y conclusiones

El objetivo principal de este trabajo se basaba en la búsqueda y aplicación de diferentes metodologías para la reducción parámetros en modelos dinámicos, como continuación de la investigación previamente hecha en [2]. La aplicación de estos métodos fue hecha en dos tipos de sistemas multicuerpos, uno de alta movilidad y otro de baja movilidad para establecer patrones de comportamiento y su aplicabilidad en diferentes sistemas. Los diferentes métodos aplicados en este trabajo son:

1. QR Selection.
2. Backward Elimination.
3. Foward Selection.
4. Foward Selection Iterativo.
5. Parametros Esenciales.
6. Norma L1.
7. Norma L1 Iterativo.
8. LASSO.

Los primeros tres métodos fueron establecidos ya en [2] el cuarto método fue presentado como una propuesta. Sin embargo como contribución, en este trabajo este metodo fue implementado y completado. Arrojando resultados interesantes, ya que se ha caído en cuenta de que el método presentado estaba incompleto y que para funcionar necesitaba de ciertas condiciones que fueron agregadas en este trabajo. Por otra parte, el método de parámetros esenciales, fue planteado en [9]. Los métodos presentados como innovadores en el área y de los cuales se hizo foco en este trabajo han sido el 6, 7,8.

En el método 6, (la norma L1) como se ha explicado anteriormente, su uso solo se limitaba al contexto de señales. Básicamente estaba orientado en la compresión de señales, en la cual, al ser aplicada se obtenían soluciones dispersas. Por lo que muchos de los coeficientes usados para representar esta señal se hacían 0 reduciendo el número de elementos que se tenían para reproducir una señal. La contribución principal de este trabajo fue llevar este mismo concepto a la DSM, para la reducción de modelos, en donde el criterio, la programación y el desarrollo han sido expuestos en este trabajo. En líneas generales, se ha propuesto una nueva metodología en la cual es posible resolver este tipo de problemas. Estas, se han comparado con las técnicas de reducción de modelos actualmente utilizadas, dando una perspectiva de las ventajas, e inconvenientes, que pueden ser presentados a la hora de aplicar este método para la reducción de modelos.

El método 7, es una extensión de la aplicación de la norma L1, su inspiración proviene del método 2 (Eliminación regresiva). En donde la idea principal pasa por aplicar la norma L1 e ir retirando el parámetro con el menor valor de manera progresiva. La intención es ir quitando en cada iteración el parámetro con la menor contribución al modelo. Aprovechando la propiedad de la norma L1, de hacer pequeños los valores que sean menos importantes al modelo. Su uso, arrojo resultados interesantes, en los cuales se llega a ver, hasta una mejora con respecto a la norma L1 solamente. Igualmente que en el caso anterior, se ha propuesto una nueva metodología en la cual, se puede resolver el problema de reducción de parámetros. Esta también ha sido comparada, con otras técnicas de reducción de modelos con la idea de ver las bondades que puede aportar al campo.

El método 8, proviene de la búsqueda de bibliografía disponible, en ella encontramos el método LASSO. El cual es un método de análisis de regresión capaz de realizar una selección de variable partiendo de los modelos estadísticos que este produce. En Matlab Optimization toolbox se tiene una función para resolver este tipo de problemas denominada *lasso*. Su implementación en el contexto de la dinámica de sistemas multicuerpo fue presentada en este trabajo. Sin embargo sus resultados quedan fuera de contexto debido a que los parámetros obtenidos, para el nivel de error deseado eran aleatorios, por tanto no podía ser comparado, con los otros métodos.

En los resultados obtenidos de la aplicación de estos métodos en el robot PUMA. Es notable ver, que el método más eficiente a la hora de reducir los parámetros es el Foward Selection iterativo. Durante la investigación hecha en este trabajo, se ha determinado que este método es cuanto menos, uno de los que presenta mayor eficiencia a la hora de ir retirando diferentes parámetros. En el caso del PUMA, ha sido solo ligeramente mejor en un punto (cuando tenemos 8 parametros) que su homónimo FS. Este método, en líneas generales, el problema que presenta, es que, a diferencia del FS se tienen soluciones en un número de parámetros N, que pueden no estar contenidas en N+1.

En el método FS con 8 parametros el error obtenido es de 0.2414 y en el FS iterativo el error obtenido es 0.237, la diferencia entrte estos dos en este punto es minima, y solo se da para este punto, de resto el comportamiento del FS con el FS iterativo es el mismo. Por lo que para este caso no hay una mejora apreciable.

El siguiente método con mayor eficiencia para reducir los parámetros del PUMA es el FS (Foward Selection). Como hemos dicho anteriormente, el solo hacer una iteración ha mejorado los resultados. La diferencia con su versión iterativa en este caso, solo se da a partir del parámetro 8.Por tanto, es casi imperceptible la diferencia que hay en la gráfica de estos dos métodos. Este método, es solo ligeramente peor que en su version iterativa. Por otra parte el

BE demuestra que es bastante bueno en el PUMA, sin embargo, no lo es más, que el FS y el FS iterativo. El método de parámetros esenciales en este caso ha quedado de cuarto lugar, sin embargo, el problema presentado con este método es que no es posible saber el orden en el que se reducen los parámetros. Su uso en este caso fue para establecer una línea de referencia ya que se establece como el estándar de facto en la reducción de modelos de la robotica industrial. Sin embargo, y como caso curioso, es interesante ver que reducir mediante los parámetros esenciales no es lo más eficiente que se puede hacer en los sistemas dinámicos. En quinto y sexto se tienen la norma L1 iterativo y la L1, respectivamente. En este caso la norma L1 a pesar de ser bastante rápida demostró no ser totalmente efectiva, y en su versión iterativa solo logró mejorar un poco, más no lo suficiente para ser considerada como óptima. En este caso el método QR ha quedado como el que peor desempeño ofrece para la reducción de parámetros.

El método de LASSO, como ha sido explicado anteriormente, ha sido probado he implementado en el contexto de reducción de parámetros, en modelos sistemas multicuerpos. El problema presentado, como se ha explicado anteriormente, es que LASSO busca el conjunto de parámetros necesarios para reproducir el modelo para un error dado (usualmente se establece este error como el mínimo, por tanto el número de parámetros también los serán). Sin embargo este número de parámetros es aleatorio por lo que aún, aumentando el tamaño del error, quizás aunque la solución que encuentra en ese punto sea bastante buena. Debido a las limitaciones de las herramientas disponibles calcular el error para retirar un determinado número de parámetros uno a uno, no ha sido posible al menos, en este trabajo.

Volviendo un poco a los otros métodos es importante señalar el Ridge. El Ridge en este caso, se define como: el mínimo número de parámetros en el cual, el modelo reducido puede reproducir al modelo original sin pérdida de información. En este caso es identificado en las gráficas como el punto donde el error hace un salto importante. Es muy claro ver que hay varios métodos como el L1 y el L1 iterativo, el Ridge se establece en 40. Otros, como el QR, PE que se establece en 36 (que es bastante bueno), en otros métodos como el BE, FS y FS iterativo logran tener un Ridge de 35, en otras palabras es posible solo necesitar 35 parámetros para reproducir el sistema dinámico del PUMA, sin pérdida de información. Es importante resaltar que BE, FS y FSI fueron los métodos capaces de dar con el menor número de parámetros.

De cara al cómputo de los sistemas dinámicos para el movimiento de los robots saber que se pueden usar 35 parámetros para reproducir casi exactamente lo que hacen 60 es muy revelador. Otra cosa que llama muchísimo la atención en todos estos temas es que no hay uniformidad en los métodos en cuanto a cuál es el parámetro o cuales son los parámetros más importantes para

el sistema, haciendo un ejercicio, tomamos los primeros 10 parámetros de cada uno de los métodos y los comparamos. Teniendo el siguiente mapa:

Tabla 9 Puma: Conteo de repetición de parámetros por cada método.

Orden de import.	Norma L1 iterativa	Norma L1	FS	BE	QR	Nº de veces que se repiten				
1	Iyy2	Izz2	m3	my3	my6	2	3	1	4	3
2	Izz2	Iyy2	Izz5	m5	mx6	3	2	2	1	3
3	mx2	mx2	Izz3	Izz5	Ixy6	3	3	3	2	1
4	Ixx4	Izz1	my6	my6	Iyz6	2	2	3	3	1
5	Iyy4	Izz3	mx6	mx6	Ixz6	2	3	3	3	1
6	Ixx3	Ixx3	mz6	mz5	Izz6	2	2	2	1	1
7	Izz3	Iyy4	mx2	mz6	my5	3	2	3	2	1
8	Izz1	Ixx4	my4	Iyy5	mx5	2	2	1	1	1
9	my3	my3	Izz4	Ixx5	Iyy6	4	4	1	1	1
10	Iyz2	Iyz2	my3	Izz2	Ixx6	2	2	4	3	1

Es muy claro ver que todos los métodos difieren en cuanto a cuales deben ser los 10 parámetros predominantes o más importantes para reproducir el modelo. En este punto solo hay un parámetro que es usado por casi todos excepto el QR es el **my3**, seguidos de este existen otros que aunque no son usados por todos si lo son por la mayoría, los cuales son **mx2, Izz3, my6 y mx6**, siendo solo cinco parámetros a considerar sin embargo deja claro que no hay conclusión entre los métodos, de cuáles son los parámetros más importantes para el modelo. Esto también se debe a que no hay una solución exacta, es la combinación de parámetros la que podrá dar un menor o mayor error. En teoría debe de existir una combinación de parámetros que sea la mejor y por tanto el parámetro más importante debería al menos repetirse, de forma similar pasa con el su antítesis el parámetro menos importante sin embargo, este tampoco es capaz de reproducirse en ninguno de los métodos.

Por otra parte tenemos el Hexaglide, como un ejemplo de un sistema dinámico de baja movilidad al cual, los diferentes métodos de reducción de parámetros han sido aplicados. Este caso, es un ejemplo curioso, dado que estamos en vista de un sistema dinámico con baja movilidad el cual posee variables dependientes e independientes. Los resultados mostrados obtenidos para cada método han sido interesantes, puesto que la mayoría de los métodos, se han acercado cuanto menos a lo que se establecería como una solución óptima. De hecho se podría establecer, el FS como el que establece una reducción de parámetros más eficiente de entre todos los métodos. En este caso la norma L1 iterativa logra dar con una solución que en ciertos puntos logra

mejorar a todos los métodos. Por lo que, dependiendo del número de tolerancia al error requerido, la combinación dada por la L1 podría ser hasta más interesante que la FS. Su homónima la L1 en este caso queda por detrás. Siendo bastante buena para este tipo de aplicaciones, es de notar que para hallar la solución con método Primal-Dual, ha sido una de las partes con mayor reto. La solución final es bastante buena aunque, llevó considerablemente más tiempo hallarla que en el caso del PUMA. Como resumen en el Hexaglide, el algoritmo de FSiterativo es el que segundo más óptimo en cuanto a la eficiencia en la que se reducen los parámetros. El tercero pasa hacer el L1 iterativo y el cuarto el L1, el BE queda por debajo de estos algoritmos, en penúltimo lugar los parámetros esenciales y por último el método QR.

Para el Hexaglide, el Ridge, se establece en 64 según los métodos de PE y QR, 60 para BE y 58 en los métodos de FS, FS iterativo, L1, L1 iterativo. Lo que quiere decir que es posible reproducir el Hexaglide con tan solo 58 parámetros de los 70 que se compone este. Es bastante notable la diferencia con el PUMA el cual solo necesita 35 de los 60 parametros.

Un aspecto interesante al resaltar con el método de FS iterativo, en el caso del Hexaglide, si nos fijamos en la gráfica comparativa. Hay puntos donde lo hace peor que su homóloga FS, la explicación a esto, proviene del mismo método. La solución para un número de parámetros N, no tiene por qué estar contenida en N+1, por tanto en N+2 estamos en vista de otra combinación de parámetros que tendrá otro comportamiento. Este podrá ser más o menos eficiente, sin embargo y lo que sí es de notar, es que este método primero tiene que mejorar al método de FS. Que es lo que pasa en este caso.

Una manera fácil de explicar este fenómeno es de la siguiente manera:

Supongamos un vector de 70 parámetros:

P=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70].

Probamos parámetro a parámetro en el modelo, y selecciono el que menor error introduce en el modelo. En este caso supongamos que es 1 por tanto nuestro vector queda como:

φ_R= [1].

En este caso nuestro vector de parámetros queda:

P=[2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70].

Con φ_R= [1, x] Se prueba o parámetro a parámetro, e igualmente que en el caso anterior selecciono el que menor error me introduzca al modelo en este caso supongamos que sea el 10, por tanto:

φ_R= [1, 10]

P=[2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70]

A φ_R le quitamos el vector introducido anteriormente en este caso 1 por tanto φ_R y P quedan:

φ_R = [10]

P=[1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70]

Probando diferentes parámetros y estos al ser 2 queda nuevamente 1 (puede ser que no) supongamos que sí. Por tanto:

φ_R= [1, 10].

P=[2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70].

En la tercera iteración pasa lo siguiente, Con φ_R= [1,10, x] probamos parámetro a parámetro, supongamos que este caso tengamos que sea 24.Por tanto:

φ_R= [1, 10,24]

P=[2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20 21 22 23 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70].

A φ_R le quitamos el vector introducido anteriormente en este caso 10 por tanto φ_R y P quedan:

φ_R= [1, 24]

P=[2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70].

Probando diferentes parámetros tenemos el caso que la combinación de parámetros que menor error introduce al modelo en este caso tenemos:

φ_R= [1, 24,61]

P=[2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 62 63 64 65 66 67 68 69 70].

En este punto, para N= 3 parámetros, tenemos que el error de φ_R= [1, 24,61] es menor, que el error de φ_R= [1, 10,24], por tanto, se escoge para este caso la solución φ_R= [1, 24,61]. Sin embargo, para n=2 parámetros, el error no es el mismo e incluso se prueba que es peor ya que la combinación más eficiente para n=2 es φ_R= [1, 10]. Por tanto el modelo obtenido de FS deja de reproducirse, lo que hace el comportamiento de este sea cuanto menos distinto ya que de ahora en adelante se parte de φ_R= [1, 24,61]. Y no de φ_R= [1, 10,24].

De la misma manera que en el Puma se hizo un análisis de los primeros diez parámetros más importantes para el modelo considerado por cada método, y se obtuvieron los siguientes resultados:

Tabla 10 Hexaglide: Conteo de repetición de parámetros por cada método.

Orden de import.	Norma L1 iterativa	Norma L1	FS	BE	QR	Nº de repeticiones				
1	mP	mP	mP	m5	IxzP	3	3	3	4	1
2	m4	m4	IzzP	mz5	IxyP	4	4	2	1	1
3	mz1	m1	mz2	myP	IxxP	4	3	2	2	1
4	m5	m2	m5	m6	IyzP	4	2	4	4	1
5	m6	m5	m4	Iyy5	mxP	4	4	4	1	1
6	m2	m6	mz1	mz1	mzP	2	4	4	4	3
7	mz4	m3	mzP	m4	myP	2	2	3	4	2
8	mz3	mz4	m6	Ixx1	IzzP	2	2	4	1	2
9	m3	mz1	mz3	mzP	IyyP	2	4	2	3	1
10	m1	mz2	m1	Ixx5	my3	3	2	3	1	1

Igualmente que en el caso anterior, no es posible establecer, una referencia clara de cuáles son, los 10 parámetros más importantes para el modelo. Sin embargo en este caso a diferencia del caso anterior existen 4 parámetros que se repiten en 4 de los métodos. Los parámetros m5 y m6, mz1, m4 son repetidos por todos los métodos. Los parámetros mP, m1, son repetidos por la mayoría (3 de los 5 métodos). En este caso quizás existe una relación más clara, por lo que muchos de los métodos pueden presentarse como óptimos.

Otro aspecto interesante, es que para reproducir el PUMA con un error menor a un 1% (un error que puede ser aceptable en la industria) solo se necesitan 18 de 60 parámetros. Mientras que en el caso de Hexaglide es posible hacerlo con 17 de 70 parámetros. Como se muestra en las gráficas 11 y 22 es casi imperceptible el error de las fuerzas obtenidas por el modelo reducido y las fuerzas reales(o del modelo entero).

Trabajos futuros:

En líneas generales, este trabajo representa una aportación al trabajo sobre reducción de parámetros presentado en [2]. Dentro de este campo se han logrado proponer tres métodos tales como: la reducción de parámetros bajo la norma L1, norma L1 iterativo y LASSO, que son capaces de ofrecer buenos resultados en la manera que se reducen los parámetros. Sin embargo es muy notable que el método más eficiente para la reducción de parámetros en cualquiera de los casos termina siendo el FS. Donde, en el caso del PUMA hace posible trabajar el modelo con solo 35 parámetros de 60 y en el Hexaglide con 58 de 70. En trabajos futuros es posible indagar en los métodos de LASSO, para la reducción de parámetros. Otra línea abierta para esta investigación, es el implementar un metodo FS iterativo con un mayor número de iteraciones, y determinar la diferencia entre la solución obtenida con un mayor número de iteraciones y la solución directa de FS. Con esto sería posible saber si vale la pena la carga computacional con respecto a la solución obtenida. También es posible que exista un número de iteraciones en donde se asegura que la solución sea lo suficientemente buena. Otra línea de investigación, es dar con un nuevo método que provenga de la combinación de los presentados en este trabajo. Por ejemplo del tipo FS aplicando la norma L1 con algún otro criterio de eliminación de los parámetros. Y por último y no menos importante, determinar el número de operaciones que se establece del resultado de cada uno de los métodos.

Bibliografía:

[1] Emmanuel Cand`es and Justin Romberg, *L1-magic : Recovery of Sparse Signals via Convex Programming,* Caltech, 2005.

[2] J. Ros, X. Iriarte, A. Paza y V. Mata, *Simplification of multibody models by parameter reduction,* 2017.

[3] J. Calzadilla, M. Vallés, V. Mata, M. Diaz-Rodriguez y A. Valera, «Adaptative control of a 3-dof parallel manipulator considering playload handling and relevant parameter models,» *Robotics and Computer-Integrated Manufacturing 30,* pp. 468-477, 2014.

[4] M. Powell, E. Cousineau y A. Ames, «Model predictive control of underactuated bipedal robotic walking,» *IEEE International conference on robotics and automation,* pp. 5121-5126, 2015.

[5] R. Reinhart, Z. Shareef y J. Steil, «Hybrid analitycal and data driven modeling for feed-foward robot control,,» *Sensors,* p. 311, 2017.

[6] E.Sanjurjo, M.A.Naya, J.-M. J.L. Blanco-Claraco y A.Gimenez-Fernandez, «Accuracy and efficiency comparison of various nonlinear kalman filters applied to multibody models,» *Nonlinear Dynamics,* pp. 1-17, 2017.

[7] G.Charles, R.Goodall y R.Dixon, «Model-based condition monitoring at the weel-rail interface,» *Vehicle System Dynamics 46,* 2008.

[8] R.Featherstone, Rigid Body Dynamics Algorithms, Springer, 2008.

[9] C. Pham y M. Gautier, «Essential Parameters of Robots,» *ENSM laboratoire d' Automatique, Ura CNRS,* nº 823, 1991.

[10] Dombre, W. Khalil y E., Modeling, identification and control of robots, Kogan Page Science, 2002.

[11] J. Ángeles, “Fundamentals of Robotic Mechanical Systems. Theory, Methods and Algorithms”, Springer US,, 2007.

[12] W. Khalil y J. Kleinfinger, «Minimum operations and minimum parameters of the dynamic models of tree structure robots, Journal of Robotics,» RA-3 (6), 1987, p. 517–526.

[13] Y. Sun y J. M. Hollerbach, «Observability index selection for robot calibration, in: Proceedings of the IEEE International Conference on,» 2008, pp. 831-836.

[14] G.Calafiore y M.Indri, «Experiment desing for robot dynamic calibration,» *Robotics and Automation;Proceedings; IEEE International Conference,* vol. 4, pp. 3303-3309, 1998.

[15] J.Swevers, C.Ganseman, J. Schutter y H. brussel, «Experimental robot identification using optimised periodic trajectories,» *Mechanical Systems and Signal Processing 9,* pp. 165-184, 1996.

[16] K. Park, «Fourier-Based optimal excitation trajectories for the dynamic identification of robots,» *Robotica 24,* pp. 625-633, 2006.

[17] B.Amstrong, «On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics,,» *International Journal of Robotics Research,* vol. 8, pp. 362-375, 1992.

[18] M.Gautier y W.Khalil, «Exciting trajectories for identification experiments involving systems with nonlinear dynamics,» *International Journal of Robotic Research,* pp. 362-375, 1992.

[19] G. J. Wiens, Shamblin, S. A. y Y. H. Oh, «Characterization of pkm dynamics in terms of system identification, Proceedings of the Institution of,» *Part K:Journal of Multi-body Dynamics,* vol. 1, nº 216, pp. 59-72, 2002.

[20] M.Gautier, «Numerical calculation of the base inertial parameters of robots,» *Journal of Robotic System ,* nº 8, pp. 485-506, 1991.

[21] O. Nelles, Nonlinear system identification, from classical approaches to neuronal networks and fuzzy models, Berlin : Springer, 2001.

[22] A. Miller, Subset selection in regresion, Chapman and Hall, 1990.

[23] Wikipedia, «Wikipedia,» 22 06 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Singular-value_decomposition. [Último acceso: 14 08 2018].

[24] R. Tibshirani, «“Regression Shrinkage and Selection via the lasso”.,» *Journal of the Royal Statistical Society. Series B,* nº 267–88., 1996.

[25] L. Breiman, “Better Subset Regression Using the Nonnegative Garrote”, Taylor & Francis, 1995.

[26] Stephanie, «Statistics How To,» 29 07 2017. [En línea]. Available: http://www.statisticshowto.com/ridge-regression/. [Último acceso: 14 08 2018].

[27] «Wikipedia,» 09 08 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Tikhonov_regularization. [Último acceso: 16 08 18].

[28] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Regularization_(mathematics). [Último acceso: 16 08 2018].

[29] mathworks, «mathworks,» [En línea]. Available: https://es.mathworks.com/help/stats/lasso.html. [Último acceso: 18 07 2018].

[30] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Norma_vectorial. [Último acceso: 25 07 2018].

[31] H. Nyquist, «Certain Topics in Telegraph Transmission Theory,» *Transactions of the A.I.E.E.,* pp. 617-644, 1928.

[32] C. Shannon, «Communication in the Presence of Noise,» *Proceedingd of the IRE ,* nº 37, pp. 10-21, 1949.

[33] E. Candès, J. Romberg y T. T., «Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,» Submitted to IEEE Trans. Inform., 2004.

[34] E. Candès, J. Romberg y T. Tao, «Near-optimal signal recovery from random projections and universal,» submitted to IEEE Trans. Inform. Theory,, 2004.

[35] Nemirovski, N. Y. E y A. S., «Interior Point Polynomial Methods in Convex Programming.,» *SIAM Publications,* 1994.

[36] Vandenberghe, S. Boyd y L., Convex Optimization., Cambridge University Press, 2004.

[37] C. Departamento de Matematicas, 21 abril 2010. [En línea]. Available: http://www.mty.itesm.mx/etie/deptos/m/ma00-130/lecturas/m130-16.pdf. [Último acceso: 14 08 2018].

[38] J. R. Shewchuk., An introduction to the conjugate gradient method without the agonizing, Manuscript, 1994.

[39] G. Meurant, The lanczos and conjugate gradient algorithms: from theory to finite precision computations, Philadelphia: SIAM,2006, 2006.

[40] V. Mata, J. Ros y X. Iriarte, «3D inertia transfer concept and symbolic determination of the base inertial parameters,» *Mechanism and machine theory 49,* pp. 284-297, 2012.

[41] E. W.Khalil, «Modeling , identification and control of robots,» *Kogan page science,* 2002.

[42] F. Benimeli, V. Mata y F. Valero, «A comparison between direct and indirect dynamic parameter identification methods in industrial robots.,» *Robotica 24,* pp. 579-590, 2006.

[43] J. Ros, L. Arrondo, J. Gil y X. Iriarte, «Lib3D_MEC_GiNaC, a library for symbolic multibody dynamics, in : Multibody Dynamics,» de *ECCOMAS Thematic Conference,*, 2007.

[44] https://es.mathworks.com/help/symbolic/cond.html, «mathworks,» [En línea]. Available: https://es.mathworks.com/help/symbolic/cond.html. [Último acceso: 14 08 2018].

[45] J. Swevers, C. Ganseman y J. Hollerbach, «Experimental robot identification using optimised periodic trajectories,» *Mechanical Systems and Signal Processing 9,* pp. 165-184, 1996.

[46] F. Valero, X. Iriarte, V. Mata y J. Ros, «Identification of dynamic parameters, in low mobility mechanical systems: application to short long arm vehicle suspension,» *Vehicle System Dynamics 0,* pp. 1-23, 2013.

[47] J. Samin y P. Fisette, Symbolic Modeling of multibody System, Netherlands: 2003, 2003.

Apéndices

Reducción de Parámetros del PUMA 560 basada en [2]

```
  This algorithm calculates the reduced model of a W*phi=tau model, with the
minimum number
%  of parameters in phib, so that: norm(Wred*phib-tau)/norm(tau) < tol
%  The algorithm removes one-by-one a column of Wred
%  The column selected each time is the one that minimises:
norm(Wi*pinv(Wi)*tau - tau)/norm(tau)
%  being Wi the matrix Wred without the i-th column and being pinv(Wi) the
pseudoinverse of Wi
%  function
[esterror_elim,esterror_add,esterror_qr,valerror_elim,valerror_add,valerror_qr
]=reduction_full_reduction;

% function reduction_full_reduction;
clear all
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
param_init;

%error_norm='norma_infinito';

error_norm='norma_2       ';

%  load matrix W and vector phi
% load W_estimation.mat;
%  load pnum.mat;
load pnum.mat;

load q_all.mat
load dq_all.mat
load ddq_all.mat
W_estimation=zeros(6*size(q_all,1),size(pnum,1));

for i=1:1:size(q_all,1)
    W_estimation((i-
1)*6+1:i*6,:)=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%
load q_all_validation.mat
load dq_all_validation.mat
load ddq_all_validation.mat
```

```
W_validation=zeros(6*size(q_all_validation,1),size(pnum,1));

for i=1:1:size(q_all_validation,1)
    W_validation((i-
1)*6+1:i*6,:)=W_modif(q_all_validation(i,:),dq_all_validation(i,:),ddq_all_val
idation(i,:));
end

%%%%%%%%%%%%%%%%%%%%%%%
%  P=randperm(60);
  P=[1:60];

%P=[10,12,13,15:60];

W=W_estimation(:,P);
%  clear W_estimation;
phi=pnum(P);
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
%  TAUMAT=vec2mat(tau,6);
% taumax=max(abs(TAUMAT));
% taumax=taumax(:);
taumax=[350;300;125;8;3;1];
sigma=kron(eye(length(W(:,1))/6),inv(diag(taumax)));
% Calculation of the weighted W and tau
tau_orig=tau;
W=sigma*W;
tau=sigma*tau;
%  Relative tolerance for the error
%  tol=0.001;
%  tol=inf
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=W;
%  The first reduced phi is the full phi
phired=phi;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;

%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ----------------------------the algorithm will eliminate one parameter at a
time -------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
sort_index_vector=zeros(1,length(phi));
for j=1:nparam-1
        error=Inf;
```

```
%       the algorithm will check the error eliminating each parameter of the
%       current reduced model
        for i=1:size(Wred,2)

%               v is the vector of the column indices without the i
                v=complementary(i,size(Wred,2));

%               W_i is matrix Wred without column i
                W_i=Wred(:,v);
%               tau_i is the part of tau that can be written in terms of W_i
                phi_i=pinv(W_i)*tau;
                tau_i=W_i*phi_i;
%               error_i is the normalized difference between tau and tau_i
                if error_norm=='norma_2       '
                                error_i=norm(tau-tau_i)/norm(tau);
                elseif error_norm=='norma_infinito'
                                error_i=max(abs(tau-tau_i))/max(abs(tau));
                else
                                disp('The norm has not been selected
properly')
                                return
                end
%               if the current error is smaller that all the previous ones
%               the index of the column is marked as the smallest-error-
column-index
                if error_i<error
                      error=error_i;
                      index=i;
                end
        end
%       if the error between full-model and smallest-error-reduced-model is
smaller than tol
%       then reduce the model eliminating column with index "index"
%       if error<tol
%               v is the vector of the columns without the index
                v=complementary(index,size(Wred,2));
%               sort_index_vector_elim
                sort_index_vector_elim(1,length(phi)+1-j)=indexvector(index);
%               Waux is the new reduced model matrix
%               phiaux is the new reduced model parameter vector
%               auxvector is the index vector of the new reduced model
                Waux=Wred(:,v);
                phiaux=phired(v);
                auxvector=indexvector(v);
                clear Wred;
                clear phired;
                clear indexvector;
                Wred=Waux;
                phired=phiaux;
                indexvector=auxvector;
        verror(nparam-j+1)=error;
%       vvariance(nparam-j+1)=max(diag(inv(Wred'*Wred)));
end
```

```
sort_index_vector_elim(1,1)=indexvector;
% sort_index_vector_elim;
verror(1)=1;
verror(nparam+1)=verror(nparam);
%  vvariance(1)=0;
esterror_elim=verror;
%  estvar_elim=vvariance;
clear indexvector;
%  figure,semilogy(esterror_elim,'r')
figure(1),semilogy(0:nparam,esterror_elim,'r')
hold on
clear verror
%  clear vvariance;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% errores en aceleraciones ELIM %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ddq_nom=max(abs(ddq_all),[],1)';
num_epsilon=zeros(6*size(q_all,1),1);
den_epsilon=zeros(6*size(q_all,1),1);
for i=size(W,2):-1:1
         W_i=W(:,sort_index_vector_elim(1:i));
         phi_i=pinv(W_i)*tau;
         param_i=zeros(size(W,2),1);
         param_i(sort_index_vector_elim(1:i),1)=phi_i;
         param_i;
%          P=[10,12,13,15:60];
         P=[1:60];
         param_i_cristiano=zeros(60,1);
         param_i_cristiano(P)=param_i;
         %Check
         %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
         %casi 0
          for j=1:size(q_all,1)
         M=M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano);
                  tau_j=tau_orig(6*(j-1)+1:6*j,1);
%                  tau_j
         Q=Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-
tau_j);
                  ddq_all_est_i(j,:)=(inv(M)*Q)';

M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano)*ddq_all(j,:)'-
Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-tau_j);
%          ddq_all_est_i(j,:)
%          ddq_all(j,:)
       num_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*(ddq_all_est_i(j,:)-
ddq_all(j,:))';
       den_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*ddq_all(j,:)';
          end
          BE_epsilon_ddq(i)=norm(num_epsilon)/norm(den_epsilon);
end
figure(2),semilogy(BE_epsilon_ddq,'r')
hold on
```

```
%  ------------------------------------------------------------------------
-------------------------------%  ----------------------------------------
-------------------------------------------------------------
%  ---------------------------the algorithm will add one parameter at a time
-------------------------------%  ----------------------------------------
-------------------------------------------------------------
%  ------------------------------------------------------------------------
-------------------------------
Wred=[];
indexvector=[];
for j=1:nparam
        error=Inf;
%       the algorithm will check the error adding one parameter to the
%       current reduced model
%       v is the vector of the column indices that have not been selected yet
        v=complementary(indexvector,nparam);
        for i=1:length(v)
%               W_i is matrix Wred with the additional column i
                W_i=[Wred,W(:,v(i))];
%               tau_i is the part of tau that can be written in terms of W_i
                phi_i=pinv(W_i)*tau;
                tau_i=W_i*phi_i;
%               error_i is the normalized difference between tau and tau_i
                if error_norm=='norma_2       '
                        error_i=norm(tau-tau_i)/norm(tau);
                elseif error_norm=='norma_infinito'
                        error_i=max(abs(tau-tau_i))/max(abs(tau));
                else
                        disp('The norm has not been selected
properly')
                        return
                end
%               if the current error is smaller that all the previous ones
%               the index of the column is marked as the smallest-error-
column-index
                if error_i<error
                      error=error_i;
                      index=i;
                end
        end
%       if the error between full-model and smallest-error-reduced-model is
still bigger than tol
%       then extend the model adding the column with index "index"
%       sort_index_vector_add
        sort_index_vector_add(1,j)=v(index);
%       parameter index
        indexvector=[indexvector,v(index)];
%       Wred is the new reduced model matrix
        Wred=W(:,indexvector);
        verror(j+1)=error;
%       vvariance(j+1)=max(diag(inv(Wred'*Wred)));
end
verror(1)=1.0;
%  vvariance(1)=0;
```

```
esterror_add=verror;
%  estvar_add=vvariance;
indexvector_add=indexvector;
clear indexvector;
clear verror
%  clear vvariance;
figure(1),semilogy(0:length(phi),esterror_add,'g')
addpath('./../');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% errores en aceleraciones ADD %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ddq_nom=max(abs(ddq_all),[],1)';
num_epsilon=zeros(6*size(q_all,1),1);
den_epsilon=zeros(6*size(q_all,1),1);
for i=size(W,2):-1:1
         W_i=W(:,indexvector_add(1:i));
         phi_i=pinv(W_i)*tau;
         param_i=zeros(size(W,2),1);
         param_i(indexvector_add(1:i),1)=phi_i;
         param_i;
%          P=[10,12,13,15:60];
        P=[1:60];
         param_i_cristiano=zeros(60,1);
         param_i_cristiano(P)=param_i;
         %Check
         %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
         %casi 0
          for j=1:size(q_all,1)


         M=M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano);
                  tau_j=tau_orig(6*(j-1)+1:6*j,1);
         Q=Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-
tau_j);
                  ddq_all_est_i(j,:)=(inv(M)*Q)';
M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano)*ddq_all(j,:)'-
Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-tau_j);

        num_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*(ddq_all_est_i(j,:)-
ddq_all(j,:))';
        den_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*ddq_all(j,:)';
          end
          FS_epsilon_ddq(i)=norm(num_epsilon)/norm(den_epsilon);
end

figure(2),semilogy(FS_epsilon_ddq,'g')



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

```
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------ QR DECOMPOSITION METHOD -------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  qr method
%  [Q,R,E] = QR(A) produces unitary Q, upper triangular R and a
%  permutation matrix E so that A*E = Q*R. The column permutation E is
%  chosen so that ABS(DIAG(R)) is decreasing.
[Q,R,E]=qr(W);
indexvector=E'*[1:size(E,1)]';
RAUX=R;
for j=1:nparam
        RAUX(:,nparam-j+1)=zeros(size(R,1),1);
        WAUX=Q*RAUX*E';
        tauredqr=WAUX*(pinv(WAUX)*tau);
        if error_norm=='norma_2       '
                        auxerror=norm(tau-tauredqr)/norm(tau);
        elseif error_norm=='norma_infinito'
                        auxerror=max(abs(tau-tauredqr))/max(abs(tau));
        else
                        disp('The norm has not been selected properly')
                        return
        end
                verror(nparam-j+1)=auxerror;
                clear Rred;clear Ered;clear QRE;
                Rred=RAUX(:,1:nparam-j);
                Ered=E(:,1:nparam-j);
                QRE=Q*Rred*(Ered)';
%               vvariance(nparam-j+1)=max(diag(inv(QRE'*QRE)));
end

verror(nparam+1)=verror(nparam);
%  vvariance(nparam+1)=vvariance(nparam);
esterror_qr=verror;
%  estvar_qr=vvariance;
sort_index_vector_qr=indexvector';
clear indexvector;
clear auxerror;
%  clear vvariance;
figure(1),semilogy(0:length(phi),esterror_qr,'b')
clear verror
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% errores en aceleraciones QR %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ddq_nom=max(abs(ddq_all),[],1)';
num_epsilon=zeros(6*size(q_all,1),1);
den_epsilon=zeros(6*size(q_all,1),1);

for i=size(W,2):-1:1
        W_i=W(:,sort_index_vector_qr(1:i));
```

```
        phi_i=pinv(W_i)*tau;
        param_i=zeros(size(W,2),1);
        param_i(sort_index_vector_qr(1:i),1)=phi_i;
        param_i;
%         P=[10,12,13,15:60];
           P=[1:60];
        param_i_cristiano=zeros(60,1);
        param_i_cristiano(P)=param_i;
        %Check
        %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
        %casi 0
        for j=1:size(q_all,1)


        M=M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano);
                tau_j=tau_orig(6*(j-1)+1:6*j,1);
        Q=Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-
tau_j);
                ddq_all_est_i(j,:)=(inv(M)*Q)';
M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano)*ddq_all(j,:)'-
Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-tau_j);

       num_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*(ddq_all_est_i(j,:)-
ddq_all(j,:))';
       den_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*ddq_all(j,:)';
        end
        QR_epsilon_ddq(i)=norm(num_epsilon)/norm(den_epsilon);
end
figure(2),semilogy(QR_epsilon_ddq,'b')
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ---------------------------------- VALIDATION OF THE REDUCTION METHODS ----
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  load Wvalidation.mat;
%  load W_validation.mat
W_val=W_validation(:,P);
clear W_validation;
W_validation=W_val;
tauval=W_validation*phi;
tau_orig_validation=tauval;
sigma_val=kron(eye(length(tauval)/6),inv(diag(taumax)));
W_validation=sigma_val*W_validation;
tauval=sigma_val*tauval;
for i=2:length(phi)+1
        %  independent and dependent parameters for each reduced model
        %  Indices of the independent parameters
        pind_elim=sort_index_vector_elim(1:i-1);
        pind_add =sort_index_vector_add(1:i-1);
```

```
        pind_qr  =sort_index_vector_qr(1:i-1);
        %  Indices of the dependent parameters
        pdep_elim=complementary(pind_elim,length(phi));
        pdep_add =complementary(pind_add ,length(phi));
        pdep_qr  =complementary(pind_qr  ,length(phi));
        %  tau estimation of the reduced models with the parameters estimated
from the estimation trajectory
        taured_elim=W_validation(:,pind_elim)*pinv(W(:,pind_elim))*tau;
        taured_add =W_validation(:,pind_add )*pinv(W(:,pind_add ))*tau;
        taured_qr  =W_validation(:,pind_qr  )*pinv(W(:,pind_qr  ))*tau;
        if error_norm=='norma_2       '
                        valerror_elim(i)=norm(tauval -
taured_elim)/norm(tauval);
                        valerror_add(i) =norm(tauval - taured_add
)/norm(tauval);
                        valerror_qr(i)  =norm(tauval - taured_qr
)/norm(tauval);
        elseif error_norm=='norma_infinito'
                        valerror_elim(i)=max(abs(tauval -
taured_elim))/max(abs(tauval));
                        valerror_add(i) =max(abs(tauval - taured_add
))/max(abs(tauval));
                        valerror_qr(i)  =max(abs(tauval - taured_qr
))/max(abs(tauval));
        else
                        disp('The norm has not been selected properly')
                        return
        end

        if true
                tau_1=tauval(1:6:end);
                tau_2=tauval(2:6:end);
                tau_3=tauval(3:6:end);
                tau_4=tauval(4:6:end);
                tau_5=tauval(5:6:end);
                tau_6=tauval(6:6:end);

                taured_add_1=taured_add(1:6:end);
                taured_add_2=taured_add(2:6:end);
                taured_add_3=taured_add(3:6:end);
                taured_add_4=taured_add(4:6:end);
                taured_add_5=taured_add(5:6:end);
                taured_add_6=taured_add(6:6:end);
        figure(3),plot(0:pi/50:2*pi,tau_1,'r',0:pi/50:2*pi,taured_add_1,'r--')
                figure(3),hold on
        figure(3),plot(0:pi/50:2*pi,tau_2,'g',0:pi/50:2*pi,taured_add_2,'g--')
        figure(3),plot(0:pi/50:2*pi,tau_3,'b',0:pi/50:2*pi,taured_add_3,'b--')
        figure(3),plot(0:pi/50:2*pi,tau_4,'m',0:pi/50:2*pi,taured_add_4,'m--')
        figure(3),plot(0:pi/50:2*pi,tau_5,'k',0:pi/50:2*pi,taured_add_5,'k--')
        figure(3),plot(0:pi/50:2*pi,tau_6,'c',0:pi/50:2*pi,taured_add_6,'c--')
                ss2=num2str(i-1);
                ss1=['Numero de parametros del modelo: '];
                ss=[ss1,ss2];
                figure(3),title(ss);
```

```
                pause
                figure(3),hold off
        end
end

valerror_elim(1)=1.0;
valerror_add(1) =1.0;
valerror_qr(1)  =1.0;
vparam=0:length(phi);
figure(1),semilogy(vparam,valerror_elim,'r--','LineWidth',2.0)
figure(1),semilogy(vparam,valerror_add ,'g--','LineWidth',2.0)
figure(1),semilogy(vparam,valerror_qr  ,'b--','LineWidth',2.0)
figure(1),legend('eliminating','adding','qr','eliminating','adding','qr')
figure(1),semilogy([0,length(phi)],[1e-5,1e-5],'k','LineWidth',0.5)
figure(1),semilogy([0,length(phi)],[1e-4,1e-4],'k','LineWidth',0.5)
figure(1),semilogy([0,length(phi)],[1e-3,1e-3],'k','LineWidth',0.5)
figure(1),semilogy([0,length(phi)],[1e-2,1e-2],'k','LineWidth',0.5)
figure(1),semilogy([0,length(phi)],[1e-1,1e-1],'k','LineWidth',0.5)
%title('REDUCCION Y VALIDACION PUMA. Lineas solidas: trayectorias estimacion
(25). Lineas a trazos: trayectoria validacion (1).')
figure(1),xlabel('Numero de parametros del modelo')
if error_norm=='norma_infinito'
        figure(1),ylabel('Error (Norma Infinito Normalizada)');
end
if error_norm=='norma_2       '
        figure(1),ylabel('Error (Norma 2 Normalizada)');
end
% Order of the parameters for different reduction methods
%  load psym.mat;
load psym;
psym_aux=psym(P);
clear psym;
psym=psym_aux;
clear psym_aux;
pind_elim_sym =psym(sort_index_vector_elim);
pind_add_sym  =psym(sort_index_vector_add);
pind_qr_sym   =psym(sort_index_vector_qr);
valerror_elim=valerror_elim(:);
valerror_add=valerror_add(:);
valerror_qr=valerror_qr(:);
save valerror_elim valerror_elim;
save valerror_add  valerror_add;
save valerror_qr   valerror_qr;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% errores en aceleraciones ELIM %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ddq_nom_validation=max(abs(ddq_all_validation),[],1)';

num_epsilon_validation=zeros(6*size(q_all_validation,1),1);
den_epsilon_validation=zeros(6*size(q_all_validation,1),1);
for i=size(W,2):-1:1
        W_i=W_validation(:,sort_index_vector_elim(1:i));
        phi_i=pinv(W_i)*tauval;
```

```
         param_i=zeros(size(W_validation,2),1);
         param_i(sort_index_vector_elim(1:i),1)=phi_i;
         param_i;
%          P=[10,12,13,15:60];
         P=[1:60];
         param_i_cristiano=zeros(60,1);
         param_i_cristiano(P)=param_i;
         %Check
         %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
         %casi 0
         for j=1:size(q_all_validation,1)
         M=M_modif(q_all_validation(j,:),dq_all_validation(j,:),ddq_all_validat
ion(j,:),param_i_cristiano);
                  tau_j=tau_orig_validation(6*(j-1)+1:6*j,1);
         Q=Q_modif(q_all_validation(j,:),dq_all_validation(j,:),ddq_all_validat
ion(j,:),param_i_cristiano,-tau_j);
                  ddq_all_validation_est_i(j,:)=(inv(M)*Q)';

       num_epsilon_validation(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom_validation))*(ddq_all_validation_est_i(j,:)-
ddq_all_validation(j,:))';
       den_epsilon_validation(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom_validation))*ddq_all_validation(j,:)';
         end
         BE_epsilon_ddq_validation(i)=norm(num_epsilon_validation)/norm(den_eps
ilon_validation);
end

figure(2),semilogy(BE_epsilon_ddq_validation,'r--')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%% errores en aceleraciones ADD %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ddq_nom_validation=max(abs(ddq_all_validation),[],1)';
num_epsilon_validation=zeros(6*size(q_all_validation,1),1);
den_epsilon_validation=zeros(6*size(q_all_validation,1),1);
for i=size(W,2):-1:1
         W_i=W_validation(:,indexvector_add(1:i));
         phi_i=pinv(W_i)*tauval;
         param_i=zeros(size(W_validation,2),1);
         param_i(indexvector_add(1:i),1)=phi_i;
         param_i;
%          P=[10,12,13,15:60];
         P=[1:60];
         param_i_cristiano=zeros(60,1);
         param_i_cristiano(P)=param_i;
         %Check
         %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
         %casi 0
         for j=1:size(q_all_validation,1)
         M=M_modif(q_all_validation(j,:),dq_all_validation(j,:),ddq_all_validat
ion(j,:),param_i_cristiano);

         tau_j=tau_orig_validation(6*(j-1)+1:6*j,1);
```

```
        Q=Q_modif(q_all_validation(j,:),dq_all_validation(j,:),ddq_all_validat
ion(j,:),param_i_cristiano,-tau_j);
                ddq_all_validation_est_i(j,:)=(inv(M)*Q)';

       num_epsilon_validation(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom_validation))*(ddq_all_validation_est_i(j,:)-
ddq_all_validation(j,:))';
       den_epsilon_validation(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom_validation))*ddq_all_validation(j,:)';
        end
        FS_epsilon_ddq_validation(i)=norm(num_epsilon_validation)/norm(den_eps
ilon_validation);
end
figure(2),semilogy(FS_epsilon_ddq_validation,'g--')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% errores en aceleraciones QR %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ddq_nom_validation=max(abs(ddq_all_validation),[],1)';

num_epsilon_validation=zeros(6*size(q_all_validation,1),1);
den_epsilon_validation=zeros(6*size(q_all_validation,1),1);
for i=size(W,2):-1:1
        W_i=W_validation(:,sort_index_vector_qr(1:i));
        phi_i=pinv(W_i)*tauval;
        param_i=zeros(size(W_validation,2),1);
        param_i(sort_index_vector_qr(1:i),1)=phi_i;
        param_i;
%         P=[10,12,13,15:60];
         P=[1:60];
       param_i_cristiano=zeros(60,1);
       param_i_cristiano(P)=param_i;
       %Check
       %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
       %casi 0
        for j=1:size(q_all_validation,1)
        M=M_modif(q_all_validation(j,:),dq_all_validation(j,:),ddq_all_validat
ion(j,:),param_i_cristiano);
                tau_j=tau_orig_validation(6*(j-1)+1:6*j,1);
        Q=Q_modif(q_all_validation(j,:),dq_all_validation(j,:),ddq_all_validat
ion(j,:),param_i_cristiano,-tau_j);
                ddq_all_validation_est_i(j,:)=(inv(M)*Q)';

       num_epsilon_validation(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom_validation))*(ddq_all_validation_est_i(j,:)-
ddq_all_validation(j,:))';      den_epsilon_validation(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom_validation))*ddq_all_validation(j,:)';
        end
        QR_epsilon_ddq_validation(i)=norm(num_epsilon_validation)/norm(den_eps
ilon_validation);
end
figure(2),semilogy(QR_epsilon_ddq_validation,'b--')
figure(2),legend('eliminating','adding','qr','eliminating','adding','qr')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%% FIN errores en aceleraciones %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ELIMINATION algorithm. ---------\n');
fprintf('-----------------------------------------------------------------\n');
disp('Code for pind_elim_sym')
for i=length(pind_elim_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_elim_sym(i)));
end
fprintf('\n\n')

for i=length(pind_elim_sym):-1:1
        fprintf('%d,',sort_index_vector_elim(i));
end
fprintf('\n\n')
fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ADDITION algorithm. ------------\n');
fprintf('-----------------------------------------------------------------\n');
disp('Code for pind_add_sym')
for i=length(pind_add_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_add_sym(i)));
end
fprintf('\n\n')
for i=length(pind_add_sym):-1:1
        fprintf('%d,',sort_index_vector_add(i));
end
fprintf('\n\n')
fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for QR algorithm. ------------------\n');
fprintf('-----------------------------------------------------------------\n');
disp('Code for pind_qr_sym')
for i=length(pind_qr_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_qr_sym(i)));
end
fprintf('\n\n')
for i=length(pind_qr_sym):-1:1
        fprintf('%d,',sort_index_vector_qr(i));
end
fprintf('\n\n')
save pind_elim_sym pind_elim_sym
save pind_add_sym  pind_add_sym
save pind_qr_sym   pind_qr_sym
save sort_index_vector_elim sort_index_vector_elim
save sort_index_vector_add  sort_index_vector_add
save sort_index_vector_qr   sort_index_vector_qr
```

Reducción de Parámetros del PUMA 560 basada en L1

```
% function reduction by L1 norm ;
clear all
% close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
%error_norm='norma_infinito';
error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
%  load pnum.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W_estimation((i-
1)*6+1:i*6,:)=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
load q_all_validation.mat
load dq_all_validation.mat
load ddq_all_validation.mat
W_validation=zeros(6*size(q_all_validation,1),size(pnum,1));
for i=1:1:size(q_all_validation,1)
    W_validation((i-
1)*6+1:i*6,:)=W_modif(q_all_validation(i,:),dq_all_validation(i,:),ddq_all_val
idation(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
   P=1:60;
%    P=[10,12,13,15:60];
%    miperm=randperm(length(P));
%    P=P(:,miperm);
 dinamic_parameters
W=W_estimation(:,P);
```

```
%  clear W_estimation;
phi=pnum(P);
% clear pnum;
 dinamic_parameters
%  W=W*PondPr;
% calculate tau in terms of W and phi
tau=W*phi;
TAUMAT=vec2mat(tau,6);
taumax=max(abs(TAUMAT));
taumax=taumax(:);
taumax=[350;300;125;8;3;1];
sigma=kron(eye(length(W(:,1))/6),inv(diag(taumax)));
% Calculation of the weighted W and tau
tau_orig=tau;
W=sigma*W;
tau=sigma*tau;
Phi0=phi;
% large scale
Afun = @(z) W*z;
Atfun = @(z) W'*z;
tic
   xp = l1eq_pd(Phi0, Afun, Atfun, tau, 1e-4, 30000, 1e-10, 200000);
% xp = l1eq_pd(Phi0, W, W', tau, 1e-6, 300, 1e-12, 200);
toc
phi=xp;
%Checks
max(abs(W*Phi0-tau))
max(abs(W*phi-tau))
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=W;
%  The first reduced phi is the full phi
phired=phi;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
sort_index_vector=zeros(1,length(phi));
phi_f=phi;
index=0;
for j=1:nparam-1
        error=Inf;
%find the minimun value:
%       v is the vector of the column indices without the j
      if index==0
          v=indexvector;
      else
                v(:,index)=[];
      end
      if index==0
         phi_f=phi_f;
      else
                 phi_f(index,:)=[];
      end
%               W_i is matrix Wred without column i
```

```
                W_i=Wred(:,v);
%               tau_i is the part of tau that can be written in terms of W_i
                phi_i=pinv(W_i)*tau;
                tau_i=W_i*phi_i;
                error_i=norm(tau-tau_i)/norm(tau);
        %               if the current error is smaller that all the previous
ones
        if error_i<error
                        error=error_i;
                                end
        [x i]=min(abs(phi_f));
    index=i;
           index=i;
   Vi= v(index);
         verror(nparam-j+1)=error;
    vindex(nparam-j+1)=Vi;
end
% sort_index_vector_elim(1,1)=indexvector;
% sort_index_vector_elim;
vindex(1,1)=v(1,2);
verror(1)=1;
verror(nparam+1)=verror(nparam);
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(1),semilogy(0:nparam,esterror_elim,'m')
hold on
% clear verror
% open GPPHI.fig
load psym;
psym_aux=psym(P);
clear psym;
psym=psym_aux;
clear psym_aux;
pind_elim_sym =psym(vindex');
disp('Code for pind_elim_sym')
for i=length(pind_elim_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_elim_sym(i)));
end
fprintf('\n\n')

for i=length(pind_elim_sym):-1:1
        fprintf('%d,',vindex(1,i));
end
fprintf('\n\n')
```

Reducción de Parámetros del PUMA 560 basada en L1 iterativo

```
% function Reduction by the iterative L1 norm;
clear all
% close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
param_init;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %error_norm='norma_infinito';
% error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
%  load pnum.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W_estimation((i-
1)*6+1:i*6,:)=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%%
load q_all_validation.mat
load dq_all_validation.mat
load ddq_all_validation.mat
W_validation=zeros(6*size(q_all_validation,1),size(pnum,1));
for i=1:1:size(q_all_validation,1)
    W_validation((i-
1)*6+1:i*6,:)=W_modif(q_all_validation(i,:),dq_all_validation(i,:),ddq_all_val
idation(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%%
   P=1:60;
%
%   P=[10,12,13,15:60];
%   miperm=randperm(length(P));
%   P=P(:,miperm);
W=W_estimation(:,P);
%  clear W_estimation;
phi=pnum(P);
% clear pnum;
```

```
% calculate tau in terms of W and phi
tau=W*phi;
taumax=[350;300;125;8;3;1];
sigma=kron(eye(length(W(:,1))/6),inv(diag(taumax)));
Wp_sigma=sigma*W;
tau=sigma*tau;

Wp=sigma*W;
   load xp
%   Phi0=phi;
 Phi0=xp;
% large scale
Afun = @(z) Wp*z;
Atfun = @(z) Wp'*z;
tic
   xp = l1eq_pd(Phi0, Afun, Atfun, tau, 1e-4, 30000, 1e-14, 200000);
% xp = l1eq_pd(Phi0, W, W', tau, 1e-6, 300, 1e-12, 200);
toc
phi=xp;
%Checks
max(abs(Wp*Phi0-tau))
max(abs(Wp*phi-tau))
k=(pinv(Wp)*tau);
max(abs(Wp*k-tau))
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=Wp;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
sort_index_vector=zeros(1,nparam);
 %  The first reduced phi is the full phi
phi_f=Phi0;
index=0;
for j=1:nparam-1
%        v is the vector of the column indices without the j
      if index==0
          v=indexvector;
      else
                v(:,index)=[];
      end

      if index==0
         phi_f=phi_f;
      else
                  phi_f(index,:)=[];
      end
        Wp_i=Wp(:,v);
        phi_norm_i=pinv(Wp_i)*tau;
        tau_i=Wp_i*phi_norm_i;
Phi0_i=phi_f;
Afun = @(z) Wp_i*z;
Atfun = @(z) Wp_i'*z;
tic
```

```
xp = l1eq_pd(Phi0_i, Afun, Atfun, tau_i, 1e-4, 30000, 1e-14, 200000);
toc
phi_N=xp;
                  error_i=norm(tau-tau_i)/norm(tau);
        [x i]=min(abs(phi_N));
        index=i;
   Vi= v(index);
        verror(nparam-j+1)=error_i;
    vindex(nparam-j+1)=Vi;
end
vindex(1,1)=v(1,1);
verror(1)=1;
verror(nparam+1)=verror(nparam);
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(3),semilogy(0:nparam,esterror_elim,'B')

% hold on
% clear verror
% open GPPHI.fig

fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ELIMINATION algorithm. ---------\n');
fprintf('-----------------------------------------------------------------\n');
load psym;
psym_aux=psym(P);
clear psym;
psym=psym_aux;
clear psym_aux;

pind_elim_sym =psym(vindex');



disp('Code for pind_elim_sym')
for i=length(pind_elim_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_elim_sym(i)));
end
fprintf('\n\n')

for i=length(pind_elim_sym):-1:1
        fprintf('%d,',vindex(1,i));
end
fprintf('\n\n')
error_norm='norma_2       ';
```

Reducción de Parámetros del PUMA 560 basada en FS iterativo

```
function FS Two iterations ;
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
error_norm='norma_2       ';
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W_estimation((i-
1)*6+1:i*6,:)=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
load q_all_validation.mat
load dq_all_validation.mat
load ddq_all_validation.mat
W_validation=zeros(6*size(q_all_validation,1),size(pnum,1));
for i=1:1:size(q_all_validation,1)
    W_validation((i-
1)*6+1:i*6,:)=W_modif(q_all_validation(i,:),dq_all_validation(i,:),ddq_all_val
idation(i,:));
End
%%%%%%%%%%%%%%%%%%%%%%%
%  P=randperm(60);
  P=1:60;
% P=[10,12,13,15:60];
W=W_estimation(:,P);
%  clear W_estimation;
phi=pnum(P);
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
taumax=[350;300;125;8;3;1];
sigma=kron(eye(length(W(:,1))/6),inv(diag(taumax)));
% Calculation of the weighted W and tau
```

```
tau_orig=tau;
W=sigma*W;
tau=sigma*tau;
phi_R=[];
phi_L=[];
indexparam=1:length(phi);
vparam=zeros(length(phi));
nparam=1:length(phi);
indexparam_1=1:length(phi);
v_i=[];
v_3=[];
for j=1:length(vparam)
 for i=1:length(indexparam);
 if phi_R==0
     v=i;
 else
          v=[v_i;indexparam(i)];
 end
   Wp_i=W(:,v);
   phi_i=pinv(Wp_i)*tau;
   tau_i=Wp_i*phi_i;
   error_i=norm(tau-tau_i)/norm(tau);
   verror_i(i)=error_i;
 end
 [x p]=min(abs(verror_i));
 index=indexparam(1,p);
 v_i(j,1)=index;
 phi_L1=v_i(end);
 clear verror_i
 phi_R=phi(v_i);
 v_1=v_i;
 verror_1=x;
  for k=1:length(indexparam);

     if length(phi_R)<=1;
       v=k;
     else
         if k==1;
          posphi_L=find(v_i(:,1)==(phi_L));
          v_i(posphi_L,:)=[];
           indexparam(end+1)=phi_L;
           indexparam=sort(indexparam);
           pos = ismember(indexparam(1,:),(v_i));
           pos=find(pos);
          indexparam(:,pos)=[];
       v=[v_i;indexparam(k)];
         else
             v=[v_i;indexparam(k)];
         end
     end
         Wp_k=W(:,v);
         phi_k=pinv(Wp_k)*tau;
         tau_k=Wp_k*phi_k;
         error_k=norm(tau-tau_k)/norm(tau);
```

```
          verror_k(k)=error_k;
     end
  [x1 p1]=min(abs(verror_k));
  index2=indexparam(1,p1);
  verror(j)=x1;
  v_i(j,1)=index2;
  phi_R=phi(v_i);
  v_2=v_i;
  verror_2=x1;
  if abs((abs(verror_2)-abs(verror_1)))< 1e-15;
  if length(v_i)<=1;
   v_i=v_1;
  verror(j)=verror_1;
  index2=v_i(end);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  else
   v_i=v_1;
  verror(j)=verror_1;
  index2=v_i(end-1);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  end
   if length(v_i)<=1;
       phi_L=v_i(end);
  else
   phi_L=v_i(end);
  end
    clear verror_k
       else
  if abs(verror_2)<abs(verror_1);
  if length(v_i)<=1;
  verror(j)=x1;
  v_i(j,1)=index2;
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  else
  v_i=v_2;
  verror(j)=verror_2;
%  index2=v_i(end-1);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
   end
  if length(v_i)<=1;
       phi_L=v_i(end);
  else
   phi_L=v_i(end-1);
  end
    clear verror_k
  else
  if length(v_i)<=1;
   v_i=v_1;
  verror(j)=verror_1;
  index2=v_i(end);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  else
   v_i=v_1;
  verror(j)=verror_1;
```

```
 index2=v_i(end-1);
 indexparam=setdiff(indexparam_1(1,:),(v_i));
 end
  if length(v_i)<=1;
      phi_L=v_i(end);
 else
  phi_L=v_i(end);
 end
   clear verror_k
 end
 end
 v_3{j}=v_i;
end
v_i(1,1)=v(1,1);
verror(nparam+1)=verror(nparam);
verror(1)=1;
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:length(phi));
save paraml1 paraml1;
figure(1),semilogy(0:length(phi),esterror_elim,'m')
% hold on
fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ELIMINATION algorithm. ---------\n');
fprintf('-----------------------------------------------------------------\n');
load psym;
pind_elim_sym =psym(v_i);
disp('Code for pind_elim_sym')
for i=length(pind_elim_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_elim_sym(i)));
end
fprintf('\n\n')

for i=length(pind_elim_sym):-1:1
        fprintf('%d,',v_i(i));
end
fprintf('\n\n')
error_norm='norma_2       ';
```

Reducción de Parámetros del PUMA 560 basada en PE

```
function SDV Decomposition (Esential Parameters);
clear all
% close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
% %error_norm='norma_infinito';
% error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
%  load pnum.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W_estimation((i-
1)*6+1:i*6,:)=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
load q_all_validation.mat
load dq_all_validation.mat
load ddq_all_validation.mat
W_validation=zeros(6*size(q_all_validation,1),size(pnum,1));
for i=1:1:size(q_all_validation,1)
    W_validation((i-
1)*6+1:i*6,:)=W_modif(q_all_validation(i,:),dq_all_validation(i,:),ddq_all_val
idation(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
P=1:60;
W=W_estimation(:,P);
%  clear W_estimation;
phi=pnum(P);
% clear pnum;
```

```
% calculate tau in terms of W and phi
tau=W*phi;
taumax=[350;300;125;8;3;1];
sigma=kron(eye(length(W(:,1))/6),inv(diag(taumax)));
Wp_sigma=sigma*W;
tau=sigma*tau;
[U,S,V] = svd(Wp_sigma);
RS=rank(S);
%Descomposicion de U
S1=S(1:length(S(1,:)),1:length(S(1,:)));
U1=U(1:length(U(1,:)),1:length(S(1,:)));
% U1=U(:,1:rank(S));
% S1=S(1:rank(S),1:rank(S));
%Modelo Ortogonal equivalente.
G=U1'*tau;
Z=V'*phi;
%Vector permutacion para ordenar de mayor a menor.
 indexvtau=1:length(G);
 index=0;
for j=1:length(G)
          if index==0
           v=indexvtau';
      else
                  v(index,:)=[];
          end
%           Wr_i=Wr(v,:);
             G_i=G(v,:);
     [x i]=max(abs(G_i));
          index=i;

   Ptaux= v(index);
   Go(j,:)=x;
   Pt(j,1)= Ptaux;
end
% Vector de permutacion para regresar.
 indexvtau=1:length(G);
 index=0;
 Pd=(zeros(length(G),1));

for j=1:length(G);
 V=indexvtau;
 k=Pt(j);
 Pd(k,1)=V(j);
end
%Modelo ortogonal reordenado
Go=G(Pt);
So=S(Pt,Pt);
Zo=Z(Pt);
n_param=length(Zo);
indexvector=1:n_param;
% reduccion del modelo.
for i=0:n_param-1;
n_param_red=n_param-i;
```

```
Go1=Go(1:n_param_red);
Go2=Go(n_param_red+1:length(Zo));
Supo1=So(1:n_param_red,1:n_param_red);
Supo2=So(n_param_red+1:length(Zo),n_param_red+1:length(Zo));
Zo1=Zo(1:n_param_red);
Zo2=Zo(n_param_red+1:n_param);
Supo_r=diag([diag(Supo1);diag(zeros(n_param-n_param_red))]);
Zo_r=pinv(Supo_r)*Go;
Z_r=Zo_r(Pd);
G_r=S1*Z_r;
error_i=norm(tau-(U1*G_r))/norm(tau);
         verror(n_param-i+1)=error_i;
%     vindex(n_param-j+1)=Vi;
end
verror(1)=1;
verror(n_param+1)=verror(n_param);
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:n_param);
save paraml1 paraml1;
figure(1),semilogy(0:n_param,esterror_elim,'m')
hold on
```

Validación de metodos para el PUMA560

```
% function Validation methods;
clear all
% close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
% %error_norm='norma_infinito';
% error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
%  load pnum.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W_estimation((i-
1)*6+1:i*6,:)=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
load q_all_validation.mat
load dq_all_validation.mat
load ddq_all_validation.mat
W_validation=zeros(6*size(q_all_validation,1),size(pnum,1));
for i=1:1:size(q_all_validation,1)
    W_validation((i-
1)*6+1:i*6,:)=W_modif(q_all_validation(i,:),dq_all_validation(i,:),ddq_all_val
idation(i,:));
end
%%%%%%%%%%%%%%%%%%%%%%%
   P=1:60;

W=W_estimation(:,P);
phi=pnum(P);
% clear pnum;
% calculate tau in terms of W and phi
```

```
tau=W*phi;
taumax=[350;300;125;8;3;1];
sigma=kron(eye(length(W(:,1))/6),inv(diag(taumax)));
Wp_sigma=sigma*W;
tau=sigma*tau;
Wp=sigma*W;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%L1 iterativo%%%%%%%%%%%%%%%%%%%%%%
%   load vindex;
%   V_N=fliplr(vindex);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FS iterativo%%%%%%%%%%%%%%%%%%%%%%
% load v_i;
% v_i=v_i';
% V_N =fliplr(v_i);
% %%%%%%%%%%%%%%%%%%%%%%%%%%Parametros Esenciales%%%%%%%%%%%%%%%%%
% load Pt
% v_i=Pt';
% V_N =fliplr(v_i);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%      FS      %%%%%%%%%%%%%%%%%%%%%%
load sort_index_vector_add.mat
 V_N =fliplr(sort_index_vector_add);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%      BE      %%%%%%%%%%%%%%%%%%%%%%
% load sort_index_vector_elim.mat
%  V_N =fliplr(sort_index_vector_elim);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%    QR     %%%%%%%%%%%%%%%%%%%%%%
% load sort_index_vector_qr.mat
%  V_N =fliplr(sort_index_vector_qr);
% %  %%%%%%%%%%%%%%%%%%%%%%%%%%%     L1      %%%%%%%%%%%%%%%%%%%%%%
% load vindexL1;
% V_N=fliplr(vindexL1);
if length(V_N(1,:))>1
    V_N=V_N;
else
    V_N=V_N';
end
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=Wp;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
sort_index_vector=zeros(1,nparam);
 %  The first reduced phi is the full phi
phi_f=phi;
index=0;

for j=1:nparam-1
%        v is the vector of the column indices without the j
      if index==0
          v=indexvector;
      else
%
       posv=find(v(1,:)==(index));
%          posv=find(v(:,1)==(index));
```

```
        v(:,posv)=[];
%             v(posv,:)=[]
%                v(:,index)=[];
      end
        Wp_i=Wp(:,v);
        phi_norm_i=pinv(Wp_i)*tau;
        tau_i=Wp_i*phi_norm_i;
                 error_i=norm(tau-tau_i)/norm(tau);
        index=V_N(:,j);
%            index=V_N(j,:);
         verror(nparam-j+1)=error_i;
end
verror(1)=1;
verror(nparam+1)=verror(nparam);
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(4),semilogy(0:nparam,esterror_elim,'B')
hold on
% clear verror
% open GPPHI.fig

fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ELIMINATION algorithm. ---------\n');
fprintf('-----------------------------------------------------------------\n');
load psym;
psym_aux=psym(P);
clear psym;
psym=psym_aux;
clear psym_aux;
pind_elim_sym =psym(V_N);
disp('Code for pind_elim_sym')
fileID = fopen('Parametros.txt','w');%%L1 iter
for i=length(pind_elim_sym):-1:1
        fprintf(fileID,'%s\n',char(pind_elim_sym(i)));
end
fclose(fileID);
fprintf('\n\n')
for i=length(pind_elim_sym):-1:1
        fprintf('%d,',V_N(i));
end
fprintf('\n\n')
```

Reducción de Parámetros del Hexaglide basada en [2]

```
This algorithm calculates the reduced model of a W*phi=tau model, with the
minimum number
%  of parameters in phib, so that: norm(Wred*phib-tau)/norm(tau) < tol
%  The algorithm removes one-by-one a column of Wred
%  The column selected each time is the one that minimises:
norm(Wi*pinv(Wi)*tau - tau)/norm(tau)
%  being Wi the matrix Wred without the i-th column and being pinv(Wi) the
pseudoinverse of Wi
%  function
[esterror_elim,esterror_add,esterror_qr,valerror_elim,valerror_add,valerror_qr
]=reduction_full_reduction;
%  function reduction_full_reduction;
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
param_init;
%  error_norm='norma_infinito';
error_norm='norma_2        ';
%  load matrix W and vector phi
% load W_estimation.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
vind=19:24;
vdep= 1:18;
nparam=size(pnum,1)
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    W_i=W(vind,:);
    W_d=W(vdep,:);
    A=dPhi_dq(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    dPhi_dq_i=A(:,vind);
        dPhi_dq_d=A(:,vdep);
    W_estimation((i-1)*6+1:i*6,:)=W_i+(-dPhi_dq_i'*inv(dPhi_dq_d'))*W_d;
end
W=W_estimation;
```

```
phi=pnum;
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
tau_orig=tau;
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=W;
%  The first reduced phi is the full phi
phired=phi;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ----------------------------the algorithm will eliminate one parameter at a
time -------------------------
%  ------------------------------------------------------------------------
-------------------------------
%  ------------------------------------------------------------------------
-------------------------------
sort_index_vector=zeros(1,length(phi));
for j=1:nparam-1
        error=Inf;
%       the algorithm will check the error eliminating each parameter of the
%       current reduced model
        for i=1:size(Wred,2)
%               v is the vector of the column indices without the i
                v=complementary(i,size(Wred,2));
%               W_i is matrix Wred without column i
                W_i=Wred(:,v);
%               tau_i is the part of tau that can be written in terms of W_i
                phi_i=pinv(W_i)*tau;
                tau_i=W_i*phi_i;
%               error_i is the normalized difference between tau and tau_i
                if error_norm=='norma_2       '
                        error_i=norm(tau-tau_i)/norm(tau);
                elseif error_norm=='norma_infinito'
                        error_i=max(abs(tau-tau_i))/max(abs(tau));
                else
                        disp('The norm has not been selected properly')
                        return
                end
%               if the current error is smaller that all the previous ones
%               the index of the column is marked as the smallest-error-
column-index
                if error_i<error
                      error=error_i;
                      index=i;
                end
        end
```

```
%       if the error between full-model and smallest-error-reduced-model is
smaller than tol
%       then reduce the model eliminating column with index "index"
%       if error<tol
%               v is the vector of the columns without the index
                v=complementary(index,size(Wred,2));
%               sort_index_vector_elim
                sort_index_vector_elim(1,length(phi)+1-j)=indexvector(index);
%               Waux is the new reduced model matrix
%               phiaux is the new reduced model parameter vector
%               auxvector is the index vector of the new reduced model
                Waux=Wred(:,v);
                phiaux=phired(v);
                auxvector=indexvector(v);
                clear Wred;
                clear phired;
                clear indexvector;
                Wred=Waux;
                phired=phiaux;
                indexvector=auxvector;
%       else
%               break;
%       end
        verror(nparam-j+1)=error;
end
sort_index_vector_elim(1,1)=indexvector;
sort_index_vector_elim;
verror(1)=1;
verror(71)=verror(70);
esterror_elim=verror;
clear indexvector;
figure,semilogy(0:70,esterror_elim,'r','LineWidth',2.0)
hold on
clear verror
ddq_nom=max(abs(ddq_all(:,vind)),[],1)'
num_epsilon=zeros(6*size(q_all,1),1);
den_epsilon=zeros(6*size(q_all,1),1);
for i=size(W,2):-1:1
        W_i=W(:,sort_index_vector_elim(1:i));
        phi_i=pinv(W_i)*tau;
        param_i=zeros(size(W,2),1);
        param_i(sort_index_vector_elim(1:i),1)=phi_i;
        param_i;
       param_i_cristiano=zeros(70,1);
       param_i_cristiano=param_i;
       %Check
       %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
       %casi 0
        for j=1:size(q_all,1)
        M=M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano);
                tau_j=tau_orig(6*(j-1)+1:6*j,1);
        Q=Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-
tau_j);
       jacob=dPhi_dq(q_all(j,:),dq_all(j,:),ddq_all(j,:));
```

```
        ganna=Gamma(q_all(j,:),dq_all(j,:),ddq_all(j,:));
        ddq_all_est_i(j,:)=inv(M)*jacob'*inv(jacob*inv(M)*jacob')*(ganna-
jacob*inv(M)*Q)+inv(M)*Q;
%M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano)*ddq_all(j,:)'-
Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-tau_j)
%           ddq_all_est_i(j,vind)
%           ddq_all(j,vind)
       num_epsilon(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom))*(ddq_all_est_i(j,vind)-ddq_all(j,vind))';
       den_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*ddq_all(j,vind)';
        end
        BE_epsilon_ddq(i)=norm(num_epsilon)/norm(den_epsilon);
end
plot(BE_epsilon_ddq,'r.')
%  ------------------------------------------------------------------------
------------------------------
%  ------------------------------------------------------------------------
------------------------------
%  ----------------------------the algorithm will add one parameter at a time
------------------------------
%  ------------------------------------------------------------------------
------------------------------
%  ------------------------------------------------------------------------
------------------------------
Wred=[];
indexvector=[];
verror(1)=1.0;
for j=1:nparam
        error=Inf;
%       the algorithm will check the error adding one parameter to the
%       current reduced model
%       v is the vector of the column indices that have not been selected yet
        v=complementary(indexvector,nparam);
        for i=1:length(v)
%               W_i is matrix Wred with the additional column i
                W_i=[Wred,W(:,v(i))];
%               tau_i is the part of tau that can be written in terms of W_i
                phi_i=pinv(W_i)*tau;
                tau_i=W_i*phi_i;
%               error_i is the normalized difference between tau and tau_i
                if error_norm=='norma_2       '
                        error_i=norm(tau-tau_i)/norm(tau);
                elseif error_norm=='norma_infinito'
                        error_i=max(abs(tau-tau_i))/max(abs(tau));
                else
                        disp('The norm has not been selected
properly')
                        return
                end
%               if the current error is smaller that all the previous ones
%               the index of the column is marked as the smallest-error-
column-index
                if error_i<error
                      error=error_i;
```

```
                        index=i;
                    end
            end
%         if the error between full-model and smallest-error-reduced-model is
still bigger than tol
%         then extend the model adding the column with index "index"
%   %     sort_index_vector_add
          sort_index_vector_add(1,j)=v(index);
%         parameter index
          indexvector=[indexvector,v(index)];
%         Wred is the new reduced model matrix
          Wred=W(:,indexvector);
          verror(j+1)=error;
end
verror(71)=verror(70);
esterror_add=verror;
indexvector_add=indexvector;
clear indexvector;
%  semilogy(verror,'g')
clear verror
semilogy(0:70,esterror_add,'g','LineWidth',2.0)
addpath('./../');
ddq_nom=max(abs(ddq_all(:,vind)),[],1)'
num_epsilon=zeros(6*size(q_all,1),1);
den_epsilon=zeros(6*size(q_all,1),1);
for i=size(W,2):-1:1
          W_i=W(:,indexvector_add(1:i));
          phi_i=pinv(W_i)*tau;
          param_i=zeros(size(W,2),1);
          param_i(indexvector_add(1:i),1)=phi_i;
          param_i;
         param_i_cristiano=zeros(70,1);
         param_i_cristiano=param_i;
         %Check
         %max(abs(W_estimation*param_i_cristiano-inv(sigma)*W_i*phi_i)) es
         %casi 0
          for j=1:size(q_all,1)
          M=M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano);
                   tau_j=tau_orig(6*(j-1)+1:6*j,1);
          Q=Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-
tau_j);
         jacob=dPhi_dq(q_all(j,:),dq_all(j,:),ddq_all(j,:));
         ganna=Gamma(q_all(j,:),dq_all(j,:),ddq_all(j,:));
         ddq_all_est_i(j,:)=inv(M)*jacob'*inv(jacob*inv(M)*jacob')*(ganna-
jacob*inv(M)*Q)+inv(M)*Q;
%M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano)*ddq_all(j,:)'-
Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-tau_j)
%           ddq_all_est_i(j,vind)
%           ddq_all(j,vind)
        num_epsilon(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom))*(ddq_all_est_i(j,vind)-ddq_all(j,vind))';
        den_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*ddq_all(j,vind)';
          end
          FS_epsilon_ddq(i)=norm(num_epsilon)/norm(den_epsilon);
```

```
end
plot(FS_epsilon_ddq,'g.')
%  ---------------------------------------------------------------------------
-------------------------------
%  ----------------------------------- QR DECOMPOSITION METHOD --------------
-------------------------------
%  ---------------------------------------------------------------------------
-------------------------------
%  ---------------------------------------------------------------------------
-------------------------------
%  qr method
%  [Q,R,E] = QR(A) produces unitary Q, upper triangular R and a
%  permutation matrix E so that A*E = Q*R. The column permutation E is
%  chosen so that ABS(DIAG(R)) is decreasing.
[Q,R,E]=qr(W);
indexvector=E'*[1:size(E,1)]';
RAUX=R;
for j=1:nparam
        RAUX(:,nparam-j+1)=zeros(size(R,1),1);
        WAUX=Q*RAUX*E';
        tauredqr=WAUX*(pinv(WAUX)*tau);
        if error_norm=='norma_2       '
                        auxerror=norm(tau-tauredqr)/norm(tau);
        elseif error_norm=='norma_infinito'
                        auxerror=max(abs(tau-tauredqr))/max(abs(tau));
        else
                        disp('The norm has not been selected properly')
                        return
        end
                verror(nparam-j+1)=auxerror;
end
verror(71)=verror(70);
esterror_qr=verror;
sort_index_vector_qr=indexvector';
clear indexvector;
clear auxerror;
semilogy(0:length(phi),esterror_qr,'b','LineWidth',2.0)
ddq_nom=max(abs(ddq_all(:,vind)),[],1)'
num_epsilon=zeros(6*size(q_all,1),1);
den_epsilon=zeros(6*size(q_all,1),1);
for i=size(W,2):-1:1
        W_i=W(:,sort_index_vector_qr(1:i));
        phi_i=pinv(W_i)*tau;
        param_i=zeros(size(W,2),1);
        param_i(sort_index_vector_qr(1:i),1)=phi_i;
        param_i;
       param_i_cristiano=zeros(70,1);
       param_i_cristiano=param_i;
        for j=1:size(q_all,1)
        M=M_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano);
                tau_j=tau_orig(6*(j-1)+1:6*j,1);
        Q=Q_modif(q_all(j,:),dq_all(j,:),ddq_all(j,:),param_i_cristiano,-
tau_j);
       jacob=dPhi_dq(q_all(j,:),dq_all(j,:),ddq_all(j,:));
```

```
        ganna=Gamma(q_all(j,:),dq_all(j,:),ddq_all(j,:));
        ddq_all_est_i(j,:)=inv(M)*jacob'*inv(jacob*inv(M)*jacob')*(ganna-
jacob*inv(M)*Q)+inv(M)*Q;
       num_epsilon(6*(j-1)+1:6*j,1)=
inv(diag(ddq_nom))*(ddq_all_est_i(j,vind)-ddq_all(j,vind))';
       den_epsilon(6*(j-1)+1:6*j,1)= inv(diag(ddq_nom))*ddq_all(j,vind)';
        end
        QR_epsilon_ddq(i)=norm(num_epsilon)/norm(den_epsilon);
end
plot(QR_epsilon_ddq,'b.')
return
%  ------------------------------------------------------------------------
------------------------------
%  ----------------------------------VALIDATION OF THE REDUCTION METHODS ----
------------------------------
%  ------------------------------------------------------------------------
------------------------------
%  ------------------------------------------------------------------------
------------------------------
%  load Wvalidation.mat;
load W_validation.mat
tauval=W_validation*phi;
for i=2:length(phi)+1
        %  independent and dependent parameters for each reduced model
        %  Indices of the independent parameters
        pind_elim=sort_index_vector_elim(1:i-1);
        pind_add =sort_index_vector_add(1:i-1);
        pind_qr  =sort_index_vector_qr(1:i-1);
        %  Indices of the dependent parameters
        pdep_elim=complementary(pind_elim,length(phi));
        pdep_add =complementary(pind_add ,length(phi));
        pdep_qr  =complementary(pind_qr  ,length(phi));
        %  tau estimation of the reduced models with the parameters estimated
from the estimation trajectory
        taured_elim=W_validation(:,pind_elim)*pinv(W(:,pind_elim))*tau;
        taured_add =W_validation(:,pind_add )*pinv(W(:,pind_add ))*tau;
        taured_qr  =W_validation(:,pind_qr  )*pinv(W(:,pind_qr  ))*tau;
        if error_norm=='norma_2       '
                valerror_elim(i)=norm(tauval - taured_elim)/norm(tauval);
                valerror_add(i) =norm(tauval - taured_add )/norm(tauval);
                valerror_qr(i)  =norm(tauval - taured_qr  )/norm(tauval);
        elseif error_norm=='norma_infinito'
                valerror_elim(i)=max(abs(tauval -
taured_elim))/max(abs(tauval));
                valerror_add(i) =max(abs(tauval - taured_add
))/max(abs(tauval));
                valerror_qr(i)  =max(abs(tauval - taured_qr
))/max(abs(tauval));
        else
                disp('The norm has not been selected properly')
                return
        end
end
valerror_elim(1)=1.0;
```

```
valerror_add(1) =1.0;
valerror_qr(1)  =1.0;
valerror_elim=valerror_elim(:);
valerror_add=valerror_add(:);
valerror_qr=valerror_qr(:);
save valerror_add valerror_add;
save valerror_elim valerror_elim;
save valerror_qr valerror_qr;
vparam=0:length(phi);
semilogy(vparam,valerror_elim,'r--','LineWidth',2.0)
semilogy(vparam,valerror_add ,'g--','LineWidth',2.0)
semilogy(vparam,valerror_qr  ,'b--','LineWidth',2.0)
legend('eliminating','adding','qr','svd','eliminating','adding','qr')
semilogy([0,70],[1e-5,1e-5],'k','LineWidth',0.5)
semilogy([0,70],[1e-4,1e-4],'k','LineWidth',0.5)
semilogy([0,70],[1e-3,1e-3],'k','LineWidth',0.5)
semilogy([0,70],[1e-2,1e-2],'k','LineWidth',0.5)
semilogy([0,70],[1e-1,1e-1],'k','LineWidth',0.5)
axis([0,70,1e-15,1e1])
title('REDUCCION Y VALIDACION HEXAGLIDE. Lineas solidas: trayectorias
estimacion (11). Lineas a trazos: trayectoria validacion (1).')
xlabel('Numero de parametros del modelo')
if error_norm=='norma_infinito'
        ylabel('Error (Norma Infinito Normalizada)');
end
if error_norm=='norma_2       '
        ylabel('Error (Norma 2 Normalizada)');
end
% Order of the parameters for different reduction methods
load psym.mat;
pind_elim_sym =psym(sort_index_vector_elim);
pind_add_sym  =psym(sort_index_vector_add);
pind_qr_sym   =psym(sort_index_vector_qr);
save sort_index_vector_elim sort_index_vector_elim;
save sort_index_vector_add sort_index_vector_add;
save sort_index_vector_qr sort_index_vector_qr;
disp('Code for pind_elim_sym')
for i=length(pind_elim_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_elim_sym(i)));
end
fprintf('\n\n')
disp('Code for pind_add_sym')
for i=length(pind_add_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_add_sym(i)));
end
fprintf('\n\n')
disp('Code for pind_qr_sym')
for i=length(pind_qr_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_qr_sym(i)));
end
fprintf('\n\n')
```

Reducción de Parámetros del Hexaglide basada en L1

```
%  function reduction by L1 norm;
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
%  error_norm='norma_infinito';
error_norm='norma_2       ';
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
vind=19:24;
miperm1=randperm(length(vind));
vind=vind(:,miperm1);
vdep=1:18;
miperm2=randperm(length(vdep));
vdep=vdep(:,miperm2);
% vdep= randperm(18);
nparam=size(pnum,1)
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    W_i=W(vind,:);
    W_d=W(vdep,:);
    A=dPhi_dq(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    dPhi_dq_i=A(:,vind);
         dPhi_dq_d=A(:,vdep);
    W_estimation((i-1)*6+1:i*6,:)=W_i+(-dPhi_dq_i'*inv(dPhi_dq_d'))*W_d;
end
W=W_estimation;
phi=pnum;
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
%  Relative tolerance for the error
%  tol=0.001;
%  tol=inf
```

```
tau_orig=tau;
Phi0=phi;
% large scale
Afun = @(z) W*z;
Atfun = @(z) W'*z;
tic
xp = l1eq_pd(Phi0, Afun, Atfun, tau, 7e-4, 1000000, 1e-14, 200000);
toc
phi=xp;
%Checks
max(abs(W*Phi0-tau))
max(abs(W*phi-tau))
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=W;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
phi_f=phi;
index=0;
for j=1:nparam-1
        error=Inf;
%find the minimun value:
%       v is the vector of the column indices without the j
      if index==0
          v=indexvector;
      else
                v(:,index)=[];
      end

      if index==0
         phi_f=phi_f;
      else
                phi_f(index,:)=[];
      end
%               W_i is matrix Wred without column i
                W_i=Wred(:,v);
%               tau_i is the part of tau that can be written in terms of W_i
                phi_i=pinv(W_i)*tau;
                tau_i=W_i*phi_i;
                error_i=norm(tau-tau_i)/norm(tau);
        %               if the current error is smaller that all the previous
ones
        if error_i<error
                        error=error_i;
                                end
        [x i]=min(abs(phi_f));
    index=i;
        Vi= v(index);
        verror(nparam-j+1)=error;
    vindex(nparam-j+1)=Vi;
end
% sort_index_vector_elim(1,1)=indexvector;
% sort_index_vector_elim;
```

```
 verror(1)=1;
 verror(nparam+1)=verror(nparam);
 esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(1),semilogy(0:nparam,esterror_elim,'m')
hold on
```

```
function reduction by Iterative L1 norm;
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
%  error_norm='norma_infinito';
error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
vind=19:24;
% miperm1=randperm(length(vind));
% vind=vind(:,miperm1);
vdep=1:18;
% miperm2=randperm(length(vdep));
% vdep=vdep(:,miperm2);
% vdep= randperm(18);
nparam=size(pnum,1)
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    W_i=W(vind,:);
    W_d=W(vdep,:);
    A=dPhi_dq(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    dPhi_dq_i=A(:,vind);
         dPhi_dq_d=A(:,vdep);
    W_estimation((i-1)*6+1:i*6,:)=W_i+(-dPhi_dq_i'*inv(dPhi_dq_d'))*W_d;
end
W=W_estimation;
phi=pnum;
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
```

```
%  Phi0=phi;
load xp
Phi0=xp;
% large scale
Afun = @(z) W*z;
Atfun = @(z) W'*z;
tic
xp = l1eq_pd(Phi0, Afun, Atfun, tau, 1e-6, 1000000, 1e-20, 40000);
toc
phi=xp;
%Checks
max(abs(W*Phi0-tau))
max(abs(W*phi-tau))
k=(pinv(W)*tau);
max(abs(W*k-tau))
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=W;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
phi_f=phi;
index=0;
for j=1:nparam-1
%        v is the vector of the column indices without the j
      if index==0
          v=indexvector;
      else
                v(:,index)=[];
      end
      if index==0
         phi_f=phi_f;
      else
                  phi_f(index,:)=[];
      end
%                 W_i is matrix Wred without column i
                  W_i=Wred(:,v);
%                 tau_i is the part of tau that can be written in terms of W_i
                  phi_i=pinv(W_i)*tau;
                  tau_i=W_i*phi_i;
Phi0_i=phi_f;
Afun = @(z) W_i*z;
Atfun = @(z) W_i'*z;
tic
xp = l1eq_pd(Phi0_i, Afun, Atfun, tau_i, 7e-4, 30000, 1e-14, 20000);
toc
phi_N=xp;
    error_i=norm(tau-tau_i)/norm(tau);
        [x i]=min(abs(phi_N));
    index=i;
        Vi= v(index);
        verror(nparam-j+1)=error_i;
    vindex(nparam-j+1)=Vi;
end
```

```
verror(1)=1;
verror(nparam+1)=verror(nparam);
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(1),semilogy(0:nparam,esterror_elim,'m')
```

Reducción de Parámetros del Hexaglide basada en FS iterativo

```
  function reduction by Iterative FS;
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
%  error_norm='norma_infinito';
error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
vind=19:24;
miperm1=randperm(length(vind));
vind=vind(:,miperm1);
vdep=1:18;
miperm2=randperm(length(vdep));
vdep=vdep(:,miperm2);
% vdep= randperm(18);
nparam=size(pnum,1)
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    W_i=W(vind,:);
    W_d=W(vdep,:);
    A=dPhi_dq(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    dPhi_dq_i=A(:,vind);
         dPhi_dq_d=A(:,vdep);
    W_estimation((i-1)*6+1:i*6,:)=W_i+(-dPhi_dq_i'*inv(dPhi_dq_d'))*W_d;
end
W=W_estimation;
phi=pnum;
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
phi_R=[];
```

```
phi_L=[];
indexparam=1:length(phi);
vparam=zeros(length(phi));
nparam=1:length(phi);
indexparam_1=1:length(phi);
v_i=[];
% v_i=zeros(1,length(phi));
% verror=zeros(1,length(phi));
% verror_i=zeros(1,length(phi));
% verror_k=zeros(1,length(phi));
 v_3=[];
for j=1:length(vparam)
 for i=1:length(indexparam);
 if phi_R==0
     v=i;
 else
          v=[v_i;indexparam(i)];
 end
   Wp_i=W(:,v);
   phi_i=pinv(Wp_i)*tau;
   tau_i=Wp_i*phi_i;
   error_i=norm(tau-tau_i)/norm(tau);
   verror_i(i)=error_i;
 end
 [x p]=min(abs(verror_i));
 index=indexparam(1,p);
 v_i(j,1)=index;
 phi_L1=v_i(end);
 clear verror_i
 phi_R=phi(v_i);
 v_1=v_i;
 verror_1=x;
  for k=1:length(indexparam);
     if length(phi_R)<=1;
       v=k;
     else
         if k==1;
          posphi_L=find(v_i(:,1)==(phi_L));
          v_i(posphi_L,:)=[];
           indexparam(end+1)=phi_L;
           indexparam=sort(indexparam);
           pos = ismember(indexparam(1,:),(v_i));
           pos=find(pos);
          indexparam(:,pos)=[];
       v=[v_i;indexparam(k)];

         else
             v=[v_i;indexparam(k)];
         end
     end
         Wp_k=W(:,v);
         phi_k=pinv(Wp_k)*tau;
         tau_k=Wp_k*phi_k;
         error_k=norm(tau-tau_k)/norm(tau);
```

```
          verror_k(k)=error_k;
     end
  [x1 p1]=min(abs(verror_k));
  index2=indexparam(1,p1);
  verror(j)=x1;
  v_i(j,1)=index2;
  phi_R=phi(v_i);
  v_2=v_i;
  verror_2=x1;
  if abs((abs(verror_2)-abs(verror_1)))< 1e-15;
  if length(v_i)<=1;
   v_i=v_1;
  verror(j)=verror_1;
  index2=v_i(end);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  else
   v_i=v_1;
  verror(j)=verror_1;
  index2=v_i(end-1);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  end
   if length(v_i)<=1;
       phi_L=v_i(end);
  else
   phi_L=v_i(end);
  end
    clear verror_k
       else
  if abs(verror_2)<abs(verror_1);
  if length(v_i)<=1;
  verror(j)=x1;
  v_i(j,1)=index2;
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  else
  v_i=v_2;
  verror(j)=verror_2;
%  index2=v_i(end-1);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
   end
  if length(v_i)<=1;
       phi_L=v_i(end);
  else
   phi_L=v_i(end-1);
  end
    clear verror_k
  else
  if length(v_i)<=1;
   v_i=v_1;
  verror(j)=verror_1;
  index2=v_i(end);
  indexparam=setdiff(indexparam_1(1,:),(v_i));
  else
   v_i=v_1;
  verror(j)=verror_1;
```

```
 index2=v_i(end-1);
 indexparam=setdiff(indexparam_1(1,:),(v_i));
 end
  if length(v_i)<=1;
      phi_L=v_i(end);
 else
  phi_L=v_i(end);
 end
   clear verror_k
 end

 end
  v_3{j}=v_i;
end
v_i(1,1)=v(1,1);
verror(nparam+1)=verror(nparam);
verror(1)=1;
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:length(phi));
save paraml1 paraml1;
figure(1),semilogy(0:length(phi),esterror_elim,'m')
fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ELIMINATION algorithm. ---------\n');
fprintf('-----------------------------------------------------------------\n');
load psym;
pind_elim_sym =psym(v_i);
disp('Code for pind_elim_sym')
for i=length(pind_elim_sym):-1:1
        fprintf('v1.push_back( &%s );\n',char(pind_elim_sym(i)));
end
fprintf('\n\n')
for i=length(pind_elim_sym):-1:1
        fprintf('%d,',v_i(i));
end
fprintf('\n\n')
error_norm='norma_2       ';
```

```
%  function reduction by Esential Parameters;
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
%  error_norm='norma_infinito';
error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
vind=19:24;
% miperm1=randperm(length(vind));
% vind=vind(:,miperm1);
vdep=1:18;
% miperm2=randperm(length(vdep));
% vdep=vdep(:,miperm2);
% vdep= randperm(18);
nparam=size(pnum,1)
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    W_i=W(vind,:);
    W_d=W(vdep,:);
    A=dPhi_dq(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    dPhi_dq_i=A(:,vind);
         dPhi_dq_d=A(:,vdep);
    W_estimation((i-1)*6+1:i*6,:)=W_i+(-dPhi_dq_i'*inv(dPhi_dq_d'))*W_d;
end
W=W_estimation;
phi=pnum;
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
```

```
[U,S,V] = svd(W);
% RS=rank(S);
S1=S(1:length(S(1,:)),1:length(S(1,:)));
U1=U(1:length(U(1,:)),1:length(S(1,:)));
G=U1'*tau;
Z=V'*phi;
 indexvtau=1:length(G);
 index=0;
for j=1:length(G)
          if index==0
           v=indexvtau';
      else
                  v(index,:)=[];
          end
%           Wr_i=Wr(v,:);
             G_i=G(v,:);
     [x i]=max(abs(G_i));
          index=i;

   Ptaux= v(index);
   Go(j,:)=x;
   Pt(j,1)= Ptaux;
end
 indexvtau=1:length(G);
 index=0;
 Pd=(zeros(length(G),1));
for j=1:length(G);
 V=indexvtau;
 k=Pt(j);
 Pd(k,1)=V(j);
end
Go=G(Pt);
So=S(Pt,Pt);
Zo=Z(Pt);
n_param=length(Zo);
indexvector=1:n_param;
for i=0:n_param-1;
n_param_red=n_param-i;
Go1=Go(1:n_param_red);
Go2=Go(n_param_red+1:length(Zo));
Supo1=So(1:n_param_red,1:n_param_red);
Supo2=So(n_param_red+1:length(Zo),n_param_red+1:length(Zo));
Zo1=Zo(1:n_param_red);
Zo2=Zo(n_param_red+1:n_param);
Supo_r=diag([diag(Supo1);diag(zeros(n_param-n_param_red))]);
Zo_r=pinv(Supo_r)*Go;
Z_r=Zo_r(Pd);
G_r=S1*Z_r;
error_i=norm(tau-U1*G_r)/norm(tau);
         verror(n_param-i+1)=error_i;
end
verror(1)=1;
verror(nparam+1)=verror(nparam);
esterror_elim=verror;
```

```
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(1),semilogy(0:nparam,esterror_elim,'m')
```

Validación de metodos para el Hexaglide

```
% function Validation Methods Hexaglide;
clear all
% close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
addpath('./../lin_alg/');
addpath('./../../../GENERATED_files_matlab/');
path(path, './Optimization');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
clear all;
warning('OFF','all');
USER_defines;
%  Init_variables_parameters;
COORD='COORD_DEP';
SIZE='DDL';
INTEG='EULER';
LIN='INV';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
param_init;
%  error_norm='norma_infinito';
error_norm='norma_2       ';
%  load matrix W and vector phi
% load W_estimation.mat;
load pnum.mat;
load q_all.mat
load dq_all.mat
load ddq_all.mat
vind=19:24;
% miperm1=randperm(length(vind));
% vind=vind(:,miperm1);
vdep=1:18;
% miperm2=randperm(length(vdep));
% vdep=vdep(:,miperm2);
% vdep= randperm(18);
nparam=size(pnum,1)
W_estimation=zeros(6*size(q_all,1),size(pnum,1));
for i=1:1:size(q_all,1)
    W=W_modif(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    W_i=W(vind,:);
    W_d=W(vdep,:);
    A=dPhi_dq(q_all(i,:),dq_all(i,:),ddq_all(i,:));
    dPhi_dq_i=A(:,vind);
         dPhi_dq_d=A(:,vdep);
    W_estimation((i-1)*6+1:i*6,:)=W_i+(-dPhi_dq_i'*inv(dPhi_dq_d'))*W_d;
end
W=W_estimation;
phi=pnum;
% clear pnum;
% calculate tau in terms of W and phi
tau=W*phi;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%L1 iterativo%%%%%%%%%%%%%%%%%%%%%%%%
%   load vindex;
%   V_N=fliplr(vindex);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FS iterativo%%%%%%%%%%%%%%%%%%%%%%%%
load v_i;
v_i=v_i';
V_N =fliplr(v_i);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%      FS      %%%%%%%%%%%%%%%%%%%%%%
% load sort_index_vector_add.mat
%  V_N =fliplr(sort_index_vector_add);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%      BE      %%%%%%%%%%%%%%%%%%%%%%
% load sort_index_vector_elim.mat
%  V_N =fliplr(sort_index_vector_elim);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%    QR     %%%%%%%%%%%%%%%%%%%%%%
% load sort_index_vector_qr.mat
%  V_N =fliplr(sort_index_vector_qr);
% %  %%%%%%%%%%%%%%%%%%%%%%%%%%%%     L1      %%%%%%%%%%%%%%%%%%%%%%
% load vindexL1;
% V_N=fliplr(vindexL1);
% % % % %
if length(V_N(1,:))>1
    V_N=V_N;
else
    V_N=V_N';
end
%  Total number of parameters of the full model
nparam=length(phi);
%  The first reduced W is the full W
Wred=W;
%  The vector of the parameters of the reduced model
indexvector=1:nparam;
sort_index_vector=zeros(1,nparam);
 %  The first reduced phi is the full phi
phi_f=phi;
index=0;
for j=1:nparam-1
%        v is the vector of the column indices without the j
      if index==0
          v=indexvector;
      else
%       posv=find(v(1,:)==(index));
%          posv=find(v(:,1)==(index));
        v(:,posv)=[];
%             v(posv,:)=[]
%                v(:,index)=[];
      end
        Wp_i=Wred(:,v);
        phi_norm_i=pinv(Wp_i)*tau;
        tau_i=Wp_i*phi_norm_i;
                 error_i=norm(tau-tau_i)/norm(tau);
        index=V_N(:,j);
%           index=V_N(j,:);
         verror(nparam-j+1)=error_i;
```

```
end
verror(1)=1;
verror(nparam+1)=verror(nparam);
esterror_elim=verror;
clear indexvector;
save verror verror;
paraml1=(0:nparam);
save paraml1 paraml1;
figure(4),semilogy(0:nparam,esterror_elim,'B')
hold on
% clear verror
% open GPPHI.fig
fprintf('-----------------------------------------------------------------\n');
fprintf('--------- Parameter order for ELIMINATION algorithm. ---------\n');
fprintf('-----------------------------------------------------------------\n');
load psym;
pind_elim_sym =psym(V_N);
disp('Code for pind_elim_sym')
fileID = fopen('Parametros.txt','w');%%L1 iter
for i=length(pind_elim_sym):-1:1
        fprintf(fileID,'%s\n',char(pind_elim_sym(i)));
end
fclose(fileID);

fprintf('\n\n')

for i=length(pind_elim_sym):-1:1
        fprintf('%d,',V_N(i));

end
fprintf('\n\n')
```