UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

DESARROLLO DE UNA TOOLBOX E INTERFAZ GRÁFICA DE USUARIO PARA LA MODELACIÓN DINÁMICA DE ROBOTS INDUSTRIALES.

"DiBotMat"

AUTORES
Calzada, Marilyn
Ibarra, Milagros

OCTUBRE 2011

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

DESARROLLO DE UNA TOOLBOX E INTERFAZ GRÁFICA DE USUARIO PARA LA MODELACIÓN DINÁMICA DE ROBOTS INDUSTRIALES.

"DiBotMat"

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRICISTA

AUTORES
Calzada, Marilyn
Ibarra, Milagros

OCTUBRE 2011

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

CERTIFICADO DE APROBACIÓN

Los abajo firmantes, miembros del jurado, designados por el Consejo de Escuela para evaluar el Trabajo Especial de Grado titulado: **DESARROLLO DE UNA TOOLBOX E INTERFAZ GRÁFICA DE USUARIO PARA LA MODELACIÓN DINÁMICA DE ROBOTS INDUSTRIALES. "DiBotMat"**, realizado por los Brs. Marilyn C., Calzada S., portadora de la cédula de identidad V-16.912.927 y Milagros J., Ibarra O., portadora de la cédula de identidad V-16.803.079; hacemos constar que hemos revisado y aprobado dicho trabajo.

_____
Prof. Teddy Rojas
Jurado

_____
Prof. Aida Pérez
Jurado

_____
Prof. Wilmer Sanz
Tutor

Octubre 2011

DEDICATORIA

Al ser divino y supremo Dios por haberme bendecido a través de la vida.

A mis padres que me inspiran, me apoyan y son mi ejemplo a seguir.

Marilyn Calzada.

DEDICATORIA

La realización de este trabajo representa un logro el cual está dedicado a:

Dios, quién me dio fe, fortaleza y salud para culminar.

Mi madre Belkis, quién con su inagotable amor y apoyo me ha impulsado a realizar mis sueños, éste es nuestro logro.

Mi padre Héctor, quién con sus palabras ha sido fuente de sabiduría.

Mi hermana Gloribel, quién compartió sus conocimientos y ha estado en todo momento a mi lado.

Mis hermanos y mis sobrinos quienes con su alegría y presencia me han dado ánimo.

Mi prima Zoraida, quién mas que una prima ha sido una hermana, amiga y consejera en todos los aspectos de mi vida.

A Oleidys, quién con su preocupación y amor ha sido una segunda madre para mí.

Mis tías Zoraida y Bárbara, quienes aportaron su granito en esta meta culminada.

Por último, pero no menos importante a aquellas personas que a lo largo de estos años se convirtieron en amigos incondicionales.

Milagros Ibarra.

AGRADECIMIENTOS

A Dios por ser en todo momento nuestra luz y guía interna.

A nuestra alma mater, la Universidad de Carabobo, por brindarnos a través de sus profesores los conocimientos para culminar esta etapa de nuestras vidas.

A nuestro tutor, Prof. Wilmer Sanz, por su paciencia, por compartir su sabiduría y darnos palabras de aliento en el momento justo. Gracias profesor, sin usted no hubiéramos logrado esta meta.

A nuestros padres, pilares fundamentales, por su apoyo incondicional con nosotras.

A nuestros hermanos y demás familiares y amigos por siempre acompañarnos

A todos muchas gracias!!!

Marilyn y Milagros.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

RESUMEN

DESARROLLO DE UNA TOOLBOX E INTERFAZ GRÁFICA DE USUARIO PARA LA MODELACIÓN DINÁMICA DE ROBOTS INDUSTRIALES. "DiBotMat"

AUTORES: CALZADA, MARILYN
IBARRA, MILAGROS
TUTOR: WILMER SANZ
FECHA: OCTUBRE 2011

El presente trabajo de grado describe el desarrollo de una toolbox para el entorno de MATLAB, ***"DiBotMat"***, sobre la modelación dinámica de manipuladores industriales, aplicando las formulaciones de Lagrange-Euler, la cual se basa en el balance energético y la de Newton-Euler, basada en la segunda ley de Newton.

En general, estas metodologías dan a conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo, es decir, la relación matemática entre la posición de las articulaciones y sus componentes derivadas: velocidad y aceleración, además del vínculo entre las fuerza y pares aplicados en los extremos del robot, y parámetros como la longitud, masas e inercias de los componentes del manipulador.

La toolbox comprende una interfaz gráfica de usuario donde se apreciará los resultados numéricos o simbólicos de las ecuaciones obtenidas por los métodos antes señalados. Estas herramientas desarrolladas se focalizan en robots planares con menos de tres grados de libertad.

Palabras claves: robots industriales, modelación dinámica, toolbox, formulaciones, pares y fuerzas.

INTRODUCCIÓN

La dinámica del robot trata no sólo de la geometría del movimiento sino de las causas que lo originan: fuerzas y momentos. El modelo dinámico de un robot se puede obtener de leyes físicas como las leyes de Newton y la Mecánica lagrangiana; esto conduce al desarrollo de las ecuaciones de movimiento dinámico para las diversas articulaciones en términos de los parámetros geométricos e inerciales de los elementos, complicándose a medida que aumenta el número de grados de libertad.

La obtención del modelo dinámico de un robot es importante si se quiere alcanzar los siguientes fines: simulación del movimiento, diseño y evaluación de la estructura mecánica, dimensión de los actuadores, diseño y evaluación de los controladores.

El objetivo es la obtención del modelo dinámico de un Robot Industrial utilizando para ello la formulación de Lagrange-Euler y la formulación de Newton-Euler. A través de este trabajo se espera desarrollar una Toolbox e Interfaz Gráfica de Usuario en el entorno de trabajo de Matlab, donde se presenten los dos algoritmos computacionales correspondientes con cada una de los modelos antes mencionados.

Para la validación de la toolbox e Interfaz Gráfica de Usuario “DiBotMat” se hará uso de *Planar Manipulators Toolbox* (PLANMANT), elaborada por *León Zlajpah.*

El desarrollo de este Trabajo de Grado está dividido en VI capítulos, pero cabe destacar la inclusión de un apéndice de guía de cómo usar la DiBotMat en el entorno de Matlab, llevando este proyecto la siguiente estructura:

Capítulo I.- PLANTEAMIENTO DEL PROBLEMA.
Capítulo II.- BASES TEÓRICAS.
Capítulo III.- MARCO METODOLÓGICO.
Capítulo IV.- DESARROLLO DE LA TOOLBOX.

Capítulo V.- DESARROLLO DE LA INTERFAZ GRÁFICA DE USUARIO.

Capítulo VI.- VALIDACIÓN DE DIBOTMAT.

CONCLUSIONES Y RECOMENDACIONES.

Apéndice A.- MANUAL DE USUARIO DE DIBOTMAT.

Apéndice B.- FUNCIONES Y SCRIPTS DE LA TOOLBOX E INTERFAZ GRÁFICA

CAPÍTULO I

1.1. PLANTEAMIENTO DEL PROBLEMA

La educación es fuente de conocimientos y medio de difusión de las mismas, lo cual permite un mayor nivel cultural que le sirve al hombre para su desarrollo y el de la sociedad. Estas consideraciones sugieren que la enseñanza y el aprendizaje son de relevante trascendencia en la formación de los individuos y que; por lo tanto, debe tenerse especial cuidado en la forma de transmitir el conocimiento, por lo que las nuevas técnicas enseñanza-aprendizaje deben enrumbarse hacia nuevos caminos, no de forma rígida e invariable.

La robótica es la ciencia encaminada a diseñar y construir aparatos y sistemas capaces de realizar tareas propias de un ser humano; además, es la tecnología del robot y para su estudio se debe tomar en cuenta la modelación dinámica del mismo, ésta analiza cuáles serán los movimientos necesarios y cuál será la fuerza que se le aplicará para que un robot pueda ejecutar cualquier tarea que se le programe.

Los manipuladores industriales son aquellos robots que realizan movimientos repetitivos para el ensamblaje u otras actividades en la industria; para su diseño y construcción, se cuenta con algunas herramientas y programas. Hoy en día, existe un software muy utilizado en universidades y centros de investigación y desarrollo llamado MATLAB (Matrix Laboratory) que es un ambiente matemático y de visualización, sencillo de utilizar, ya que los problemas y sus soluciones están expresados justamente como están escritas. MATLAB cuenta con varias familias de soluciones para aplicaciones específicas llamadas *toolboxes*, que son colecciones de funciones utilizadas para resolver alguna clase particular de problema.

En la actualidad, MATLAB no está provisto de herramientas y funciones que permita simular la modelación dinámica de un robot industrial; por lo tanto, se decidió realizar el diseño de una toolbox que solvente la situación planteada, siendo de gran ayuda para los usuarios de

MATLAB en el área de la Robótica para que sea una herramienta de fácil manipulación y acceso.

Por último, se realizará una interfaz gráfica de usuario (GUI) en MATLAB con el propósito de representar la información de la toolbox y las acciones del manipulador.

1.2. JUSTIFICACIÓN DE LA INVESTIGACIÓN

El propósito de desarrollar una toolbox en MATLAB es facilitar el estudio de la modelación dinámica de robots industriales y proporcionarle ayuda a estudiantes, investigadores o usuarios del software para la simulación de la misma.

También, contribuye con la línea de investigación de la Universidad de Carabobo en dicha área, creando una herramienta de gran utilidad que en futuro puede ser aportada a los estudiantes de Ingeniería Eléctrica de la Universidad de Carabobo que cursen la cátedra de Robótica y Visión Industrial o cualquier usuario que desee abocarse al estudio de esta rama.

Además, esta toolbox ofrece la oportunidad de proseguir estudios en esta área adaptados al ritmo del participante. De esta forma, al penetrarse en el mundo de la modelación dinámica de manipuladores industriales se pueda brindar u ofrecer oportunidades y abrir más caminos donde esta parte de la robótica se vea más atractiva, simple y entusiasme al participante para el uso de dicha herramienta y un aprovechamiento más efectivo del tiempo.

La propuesta podría estimular a docentes, estudiantes y otras personas generando interés para abordar los procesos de enseñanza y aprendizaje de esta área, así como extendiendo su aplicación por otros medios en un futuro.

1.3. OBJETIVOS DE LA INVESTIGACION

1.3.1. Objetivo General

Desarrollar una toolbox e interfaz gráfica de usuario en MATLAB para la modelación dinámica de robots industriales.

1.3.2. Objetivos Específicos

1. Comprender las metodologías Lagrange-Euler y Newton-Euler para la elaboración del modelo dinámico de un robot industrial.
2. Desarrollar scripts y funciones para la aplicación en MATLAB de los algoritmos correspondientes a las formulaciones de Langrange-Euler y Newton-Euler.
3. Diseñar una interfaz grafica de usuario para el manejo de las funciones y comandos realizados en la programación.
4. Evaluar la toolbox en MATLAB para la modelación dinámica de manipuladores industriales con ejemplos típicos y atípicos.
5. Evaluar la interfaz grafica de usuario con ejemplos típicos y atípicos.

1.4. ALCANCE DE LA INVESTIGACIÓN

En este proyecto de investigación se desarrollará una toolbox para la modelación dinámica de los robots industriales, para lograrlo se establecerán las ecuaciones dinámicas de los mismos a través de la metodología de Euler-Langrage y se utilizará MATLAB R2008a de 32 bits.

Esto se realizará con la finalidad de observar en una interfaz gráfica en MATLAB el comportamiento de un robot planar con menos de tres grados de libertad, ante diferentes condiciones de carga y gravedad.

CAPÍTULO II

2.1. ANTECEDENTES

En diferentes partes del mundo existen estudios previos que contribuyen al desarrollo de trabajos especiales de grado en el área de la Robótica; es por esto que considerando el proyecto de grado aquí propuesto se seleccionaron y revisaron cuidadosamente los proyectos de interés que se muestran a continuación:

Eduardo A. Cirera [2003] CÁLCULO DEL MODELO DINÁMICO DE UN BRAZO ROBÓTICO DE DOS ARTICULACIONES CONTROLADO POR RED TENDONAL

La propuesta de este trabajo fue obtener el modelo dinámico de un brazo robótico de dos articulaciones controlado por una red tendonal, que ofrece ventajas en cuanto a carga y flexibilidad. Se aplicó el método de Lagrange, que plantea el problema en base a términos de energía.

El modelo sugiere una serie de ecuaciones, cuya resolución permite obtener la información necesaria, la que será enviada hasta el robot mediante tarjetas electrónicas insertadas en él. Estos circuitos envían la señal de voltaje a la fuente de potencia del robot, encargada de convertirla en la potencia adecuada que necesita cada motor para realizar la función asignada. De este proyecto se tomará como contribución las bases teóricas del modelo dinámico de un brazo robótico a través del método de Lagrange.

Josué R. Martínez, Gerardo V. Guerrero, Luis G. Vela [2007] MODELADO DEL ROBOT SCORBOT ER-V+

En este trabajo se desarrolló el proceso de modelado de un robot manipulador, teniendo por objeto de estudio el robot SCORBOT ER-V+. El modelado cinemático directo se realiza por medio del algoritmo de Denavit-Hartenberg, el inverso se obtiene por métodos geométricos y el modelo dinámico se obtiene por medio de la metodología de Euler-Lagrange.

Los resultados muestran que se obtiene un modelo dinámico que presenta un comportamiento muy cercano al comportamiento del sistema real. Así como es

posible observar el procedimiento de modelado de un robot de 3 grados de libertad.

Este trabajo se relaciona con esta investigación en la modelación dinámica de un robot SCORBOT ER-V a través de la formulación de Euler-Langrage, sin embargo, para efectos de este proyecto el modelo dinámico se estudiará a través de dos metodologías para los Robot Industriales.

León Zlajpah [2000] PLANAR MANIPULATORS TOOLBOX.

PLANMANT es un paquete de simulación para manipuladores de simulación plana de n-DOF con juntas de revolución en el cuadro de herramientas de MATLAB y Simulink. Es una herramienta muy útil y eficaz para muchos propósitos: simulación cinemática, simulación dinámica, análisis y síntesis de sistemas de control, la generación de la trayectoria, etc La caja de herramientas ha sido desarrollado para hacer más fácil la simulación y el análisis de sistemas redundantes.

El Planar Manipuladores Toolbox es basado en un modelo cinemático y dinámico de un manipulador planar con juntas de revolución. La caja de herramientas de simulación permite derivados de manipuladores con muchos grados de libertad dentro de los tiempos de simulación razonable. En realidad, el paquete permite la simulación en tiempo real y el control del robot en línea, es decir, la simulación del manipulador, utilizando bloques de funciones de MATLAB y Simulink. Esta Toolbox tiene un gran aporte en este trabajo debido a que se utilizará para la simulación de los datos generados de un robot predefinido.

Wilmer Sanz, Luis Obediente, Clemente Herrera [2004] DISEÑO DE SOFTWARE PARA LA MODELACIÓN DE MANIPULADORES CON MATLAB Y SIMULACIÓN EN 3D ROBOWORKS.

En este artículo se describe el software para la modelación de robots industriales, PGIBOT Mat, el cual se ha desarrollado mediante la interfaz gráfica de usuario (GUI) de MATLAB como una herramienta para facilitar el uso de una Toolbox de robótica y como un medio para generar archivos de simulación en el

ambiente 3D de RoboWorks. La Toolbox referida constituye un desarrollo personal para aplicar nueve tipos de transformaciones homogéneas, determinar las transformaciones que definen el movimiento relativo entre eslabones consecutivos de una cadena cinemática, realizar el estudio cinemático directo y simular el funcionamiento de un Teach Pendant en la planificación de tareas simples para Robots Industriales (RI) con seis o menos grados de libertad; los autores se enfocaron en un proceso similar al estudiado en el presente proyecto, lo cual puede servir a manera de referencia para el desarrollo del mismo. El aporte de este trabajo es la ayuda investigativa sobre la modelación de los robots en el ambiente de MATLAB.

2.2. BASES TEÓRICAS

2.2.1. ROBOTS INDUSTRIALES

2.2.1.1. Definición

Existen ciertas dificultades a la hora de establecer una definición formal de lo que es un robot industrial. La primera de ellas surge de la diferencia conceptual entre el mercado japonés y el euro-americano de lo que es un robot y lo que es un manipulador. Así, mientras que para los japoneses un robot industrial es cualquier dispositivo mecánico dotado de articulaciones móviles destinado a la manipulación, el mercado occidental es más restrictivo, exigiendo una mayor complejidad, sobre todo en lo relativo al control. En segundo lugar, y centrándose ya en el concepto occidental, aunque existe una idea común acerca de lo que es un robot industrial, no es fácil ponerse de acuerdo a la hora de establecer una definición formal. Además, la evolución de la robótica ha ido obligando a diferentes actualizaciones de su definición.

La definición más comúnmente aceptada posiblemente sea la de la Asociación de Industrias Robóticas (RIA), según la cual:

Un **Robot Industrial** (RI) es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas, o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas.

Esta definición, ligeramente modificada, ha sido adoptada por la Organización Internacional de Estándares (ISO) que define al robot industrial como:

Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.

Se incluye en esta definición la necesidad de que el robot tenga varios grados de libertad. Una definición más completa es la establecida por la Asociación Francesa de Normalización (AFNOR), que define primero el manipulador y, basándose en dicha definición, el robot:

Manipulador: mecanismo formado generalmente por elementos en serie, articulados entre sí**,** destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante dispositivo lógico.

Robot: manipulador automático servo-controlado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles y dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o de varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.

Por último, la Federación Internacional de Robótica (IFR) distingue entre robot industrial de manipulación y otros robots:

Por robot industrial de manipulación se entiende una maquina de manipulación automática, reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos

especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento.

En esta definición se debe entender que la reprogramabilidad y la multifunción se consiguen sin modificaciones físicas del robot.

Común en todas las definiciones anteriores es la aceptación del robot industrial como un brazo mecánico con capacidad de manipulación y que incorpora un control más o menos complejo. [1]

2.2.1.2. Componentes

El componente principal lo constituye el manipulador, el cual consta de varias articulaciones y sus elementos (fig. 2.1).

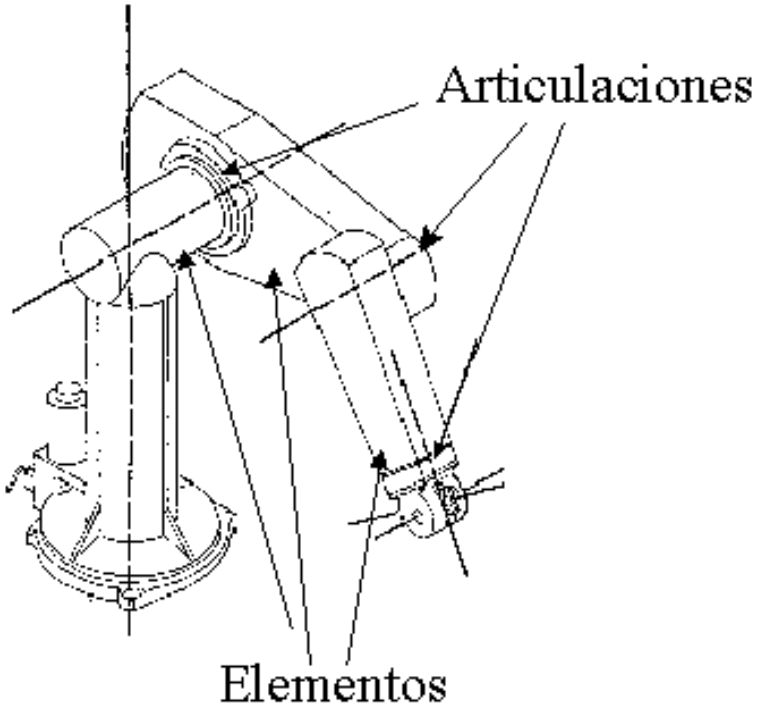


Fig. 2.1. Componentes y elementos del robot industrial. [2]

2.2.1.3. Elementos

Las partes que conforman el manipulador reciben los nombres de: cuerpo, brazo, muñeca y efector final. Al efector final se le conoce comúnmente como sujetador o gripper (fig. 2.2). [2]

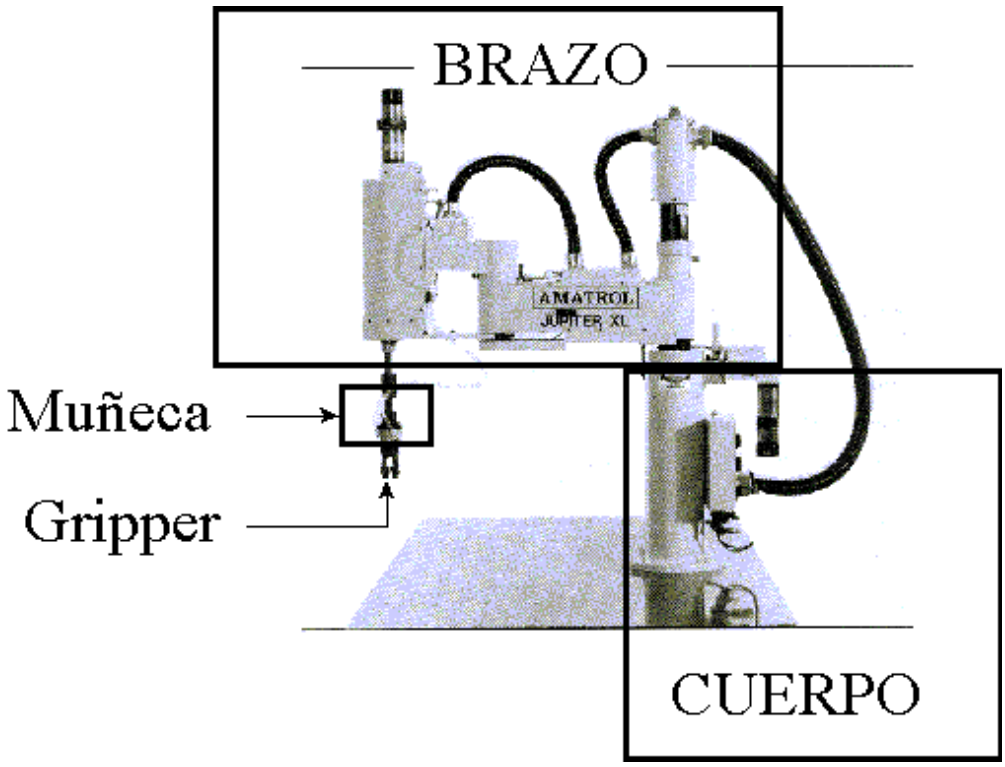


Fig. 2.2. Elementos de un robot industrial. [2]

Además del manipulador, los otros elementos que forman parte del robot son un controlador, mecanismos de entrada y salida de datos y dispositivos especiales. El controlador del robot, como su nombre lo indica, es el que controla cada uno de los movimientos del manipulador y guarda sus posiciones. El controlador recibe y envía señales a otras máquinas-herramientas (por medio de señales de entrada/salida) y almacena programas. Los mecanismos de entrada y salida, más comunes son: teclado, monitor y caja de comandos llamada "teach pendant". En el dibujo anterior tenemos un controlador (computer module) que envía señales a los motores de cada uno de los ejes del robot, la caja de comandos ("teach pendant") la cual sirve para enseñarle las posiciones al manipulador del robot. [2]

2.2.1.4. Clasificación

Tabla 2.1.- Clasificación de los robots industriales según la AFRI. [3]

	Clasificación de los Robots
Tipo A	Manipulador con control manual o telemando.
Tipo B	Manipulador automático con ciclos preajustados; regulación mediante fines de carrera o topes; control por PLC; accionamiento neumático, eléctrico o hidráulico.
Tipo C	Robot programable con trayectoria continua o punto a punto. Carece de conocimiento sobre su entorno.
Tipo D	Robot capaz de adquirir datos de su entorno, readaptando su tarea en función de éstos.

Fuente: Fundamentos de la Robótica, 1997.

La IFR distingue ente cuatro tipos de robots:

- Robot secuencial.
- Robot de trayectoria controlable.
- Robot adaptativo.
- Robot telemanipulado.

Esta clasificación coincide en gran medida con la establecida con la Asociación Francesa de Robótica Industrial (AFRI). [3]

2.2.1.5. Grados de Libertad (GDL)

Cada uno de los movimientos independientes (giros y desplazamientos) que puede realizar cada articulación con respecto a la anterior. Son los parámetros que se precisan para determinar la posición y la orientación del elemento terminal del manipulador. El número de grados de libertad del robot viene dado por la suma de los GDL de las articulaciones que lo componen. Puesto que las articulaciones empleadas suelen ser únicamente de rotación y prismáticas, con un solo grado de

libertad cada una, el número de GDL del robot suele coincidir con el número de articulaciones que lo componen.

Puesto que para posicionar y orientar un cuerpo de cualquier manera en el espacio son necesarios seis parámetros, tres para definir la posición y tres para la orientación, si se pretende que un robot posicione y oriente su extremo (y con él la pieza o herramienta manipulada) de cualquier modo en el espacio, se precisará al menos seis grados de libertad.

Fig. 2.3. Estructura de un robot de 6 grados de libertad.

En la fig. 2.3 se muestra el esquema de un robot de estructura moderna con 6 GDL; tres de ellos determinan la posición del aprehensor en el espacio (q1, q2 y q3) y los otros 3, la orientación del mismo (q4, q5 y q6).

Un mayor número de grados de libertad conlleva un aumento de la flexibilidad en el posicionamiento del elemento terminal. Aunque la mayoría de las aplicaciones industriales requieren 6 GDL, como las de la soldadura, mecanizado y paletización, otras más complejas requieren un número mayor, tal es el caso en las labores de montaje. Si se trabaja en un entorno con obstáculos, el dotar al robot de grados de libertad adicionales le permitirá acceder a posiciones y orientaciones de su extremo a las que, como consecuencia de los obstáculos, no hubieran llegado con seis grados de libertad. Otra situación frecuente es dotar al robot de un grado de libertad adicional que le permita desplazarse a lo largo de un

carril aumentando así el volumen del espacio al que puede acceder. Tareas más sencillas y con movimientos más limitados, como las de la pintura y paletización, suelen exigir 4 o 5 GDL.

Cuando el número de grados de libertad del robot es mayor que los necesarios para realizar una determinada tarea se dicen que el robot es redundante.

Observando los movimientos del brazo y de la muñeca, podemos determinar el número de grados de libertad que presenta un robot. Generalmente, tanto en el brazo como en la muñeca, se encuentra un abanico que va desde uno hasta los tres GDL. Los grados de libertad del brazo de un manipulador están directamente relacionados con su anatomía o configuración. [4]

2.2.1.6. Tipos de Articulaciones

Cada articulación provee al robot de, al menos, un grado de libertad. En otras palabras, las articulaciones permiten al manipulador realizar movimientos:

- **Lineales** que pueden ser horizontales o verticales (fig. 2.4). [4]

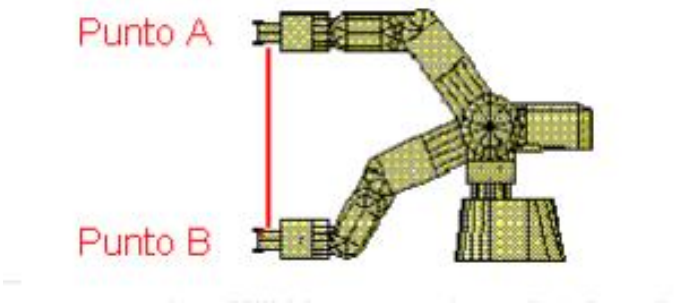


Fig. 2.4. Movimiento lineal entre los puntos A-B. [4]

- **Angulares** (por articulación) (fig. 2.5).

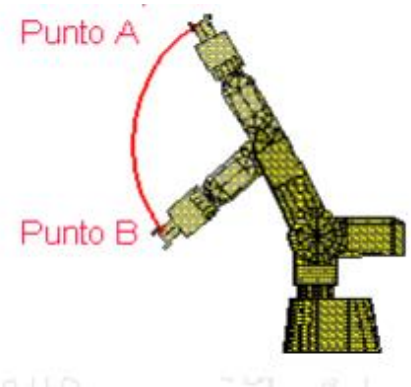


Fig. 2.5. Movimiento angular (por articulación) puntos A-B. [4]

(En los dos casos la línea roja representa la trayectoria seguida por el robot).

Existen dos tipos de articulación utilizados en las juntas del manipulador:

• Prismática /Lineal - junta en la que el eslabón se apoya en un deslizador lineal. Actúa linealmente mediante los tornillos sinfín de los motores, o los cilindros.

• Rotacional - junta giratoria a menudo manejada por los motores eléctricos y las transmisiones, o por los cilindros hidráulicos y palancas. [4]

2.2.1.7. Configuración

Se consideran, en primer lugar, las estructuras más utilizadas como brazo de un robot manipulador. Estas estructuras tienen diferentes propiedades en cuanto a espacio de trabajo y accesibilidad a posiciones determinadas. En la fig. 2.6 se muestran cuatro configuraciones básicas. [5]

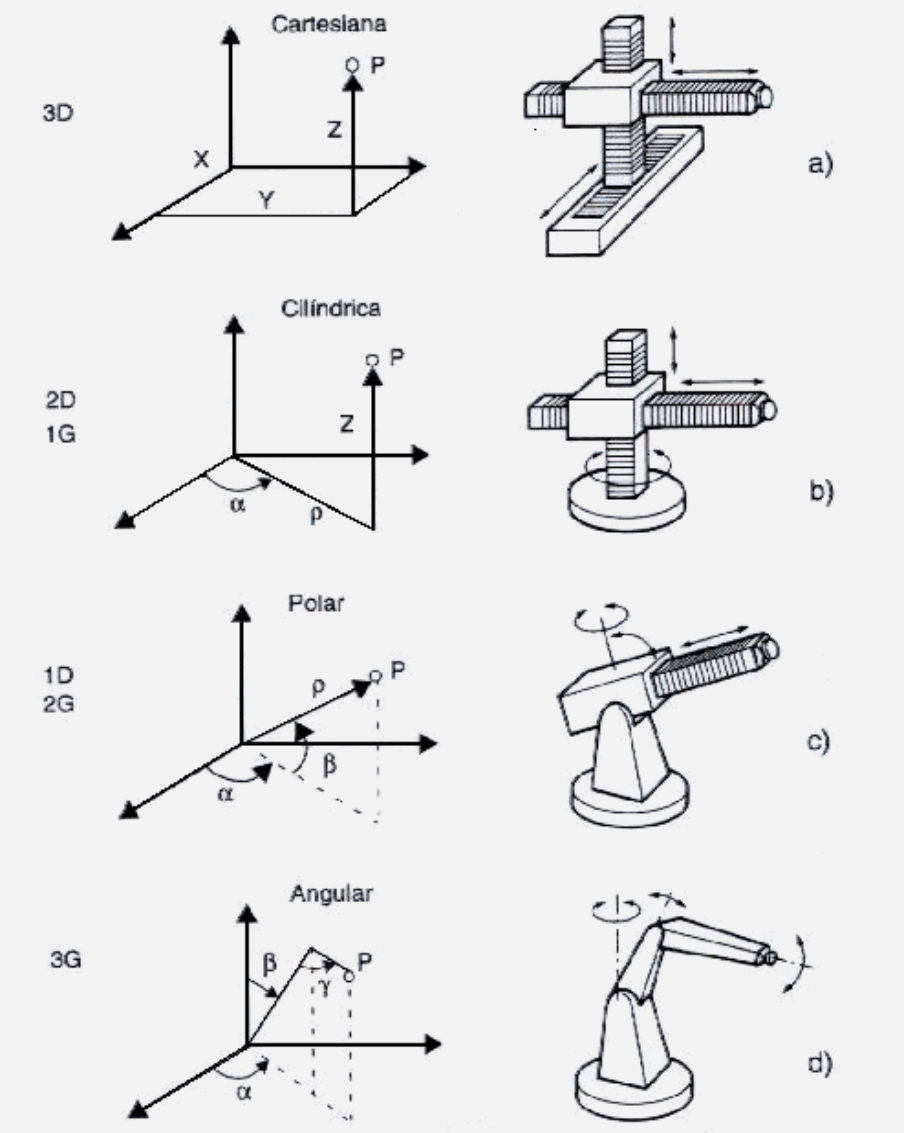


Fig. 2.6. Configuraciones básicas de los manipuladores. [5]

El espacio de trabajo es el conjunto de puntos en los que puede situarse el efector final del manipulador. Corresponde al volumen encerrado por las superficies que determinan los puntos a los que accede el manipulador con su estructura totalmente extendida y totalmente plegada.

Por otra parte, todos los puntos del espacio de trabajo no tienen la misma accesibilidad. Los puntos de accesibilidad mínima son los que las superficies que delimitan el espacio de trabajo ya que a ellos solo puede llegarse con una única orientación. [5]

- **Configuración cartesiana.**

La configuración tiene tres articulaciones prismáticas (3D o estructura PPP). Esta configuración es bastante usual en estructuras industriales, tales como pórticos, empleadas para el transporte de cargas voluminosas. La especificación de posición de un punto se efectúa mediante las coordenadas cartesianas (*x*, *y* y *z*). Los valores que deben tomar las variables articulares corresponden directamente a las coordenadas que toma el extremo del brazo. Esta configuración no resulta adecuada para acceder a puntos situados en espacios relativamente cerrados y su volumen de trabajo es pequeño cuando se compara con el que puede obtenerse con otras configuraciones. [5]

- **Configuración cilíndrica.**

Esta configuración tiene dos articulaciones prismáticas y una de rotación (2D, 1G). La primera articulación es normalmente de rotación (estructura RPP). La posición se especifica de forma natural en coordenadas cilíndricas. Esta configuración puedes ser de interés en una célula flexible, con el robot situado en el centro de la célula sirviendo a diversas máquinas dispuestas radialmente a su alrededor. El volumen de trabajo de esta estructura RPP (o de la PRP), suponiendo un radio de giro de 360 grados y un rango de desplazamiento de L, es el de un toro de sección cuadrada de radio interior L y radio exterior 2L. Se demuestra que el volumen resultante es: $3\pi L^3$. [5]

- **Configuración polar o esférica.**

Está configuración se caracteriza por dos articulaciones de rotación y una prismática (2G, 1D o estructura RRP). En este caso las variables articulares expresan la posición del extremo del tercer enlace en coordenadas polares.

En un manipulador con tres enlaces de longitud L, el volumen de trabajo de esta estructura, suponiendo un radio de giro de 360 grados y un rango de desplazamiento de L, es el que existe entre una esfera de radio 2L y otra concéntrica de radio L. Por consiguiente el volumen es: $(28/3)\pi L^3$. [5]

- **Configuración angular.**

Esta configuración es una estructura con tres articulaciones de rotación (3G o RRR). La posición del extremo final se especifica de forma natural en coordenadas angulares.

La estructura tiene un mejor acceso a espacios cerrados y es fácil desde el punto de vista constructivo. Es muy empleada en robots manipuladores industriales, especialmente en tareas de manipulación que tengan una cierta complejidad. La configuración angular es la más utilizada en educación y actividades de investigación y desarrollo. En esta estructura es posible conseguir un gran volumen de trabajo. Si la longitud de sus tres enlaces es de L, suponiendo un radio de giro de 360 grados, el volumen de trabajo sería el de una esfera de radio 2L, es decir: $(34/3)\pi L^3$. [5]

- **Configuración SCARA.**

Esta configuración está especialmente diseñada para realizar tareas de montaje en un plano. Está constituida por dos articulaciones de rotación con respecto a dos ejes paralelos, y una de desplazamiento en sentido perpendicular al plano. El volumen de trabajo de este robot, suponiendo segmentos de longitud L, un radio de giro de 360 grados y un rango de desplazamiento de L es de $4\pi L^3$.

Para llevar a cabo los cálculos y de esta forma asegurar su correcto funcionamiento del robot en cuanto a la cinemática y dinámica se refiere, se toma en consideración la teoría que tiene por objeto crear las bases de un modelo matemático del sistema. [5]

2.2.1.8. Matriz de Transformación Homogénea

Sirve para expresar la orientación y posición de un sistema de referencia móvil con respecto a otro fijo XY. [6]

Matriz 4x4, como se muestra en la fig. 2.7, que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro. [7]

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{p}_{3\times1} \\ \mathbf{f}_{1\times3} & \mathbf{w}_{1\times1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix}$$

Fig. 2.7. Matriz de transformación. [7]

En robótica la submatriz f_{1x3}, que representa una transformación de perspectiva, es nula; y la submatriz w_{1x1}, que representa un escalado global, es la unidad, como se muestra en la fig. 2.8.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{p}_{3\times1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ 0 & 1 \end{bmatrix}$$

Fig. 2.8. Matriz de transformación con perspectiva nula y escalado unitario. [7]

Esta matriz representa la orientación y posición de un sistema OUVW rotado y trasladado con respecto al sistema de referencia OXYZ y sus aplicaciones son:

- Representar la posición y orientación de un sistema girado y trasladado OUVW, con respecto a un sistema fijo de referencia OXYZ, que es lo mismo que representar una rotación y una traslación realizada sobre un sistema de referencia. [7]
- Transformar un vector expresado en coordenadas con respecto a un sistema OUVW, a su expresión en coordenadas del sistema de referencia OXYZ, como se puede observar en la fig. 2.9. [7]

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix}$$

Fig. 2.9. Transformación de un vector de un sistema OUVW a uno OXYZ. [7]

- Rotar y trasladar un vector con respecto a un sistema de referencia fijo OXYZ, como se muestra en la fig. 2.10. [7]

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix}$$

Fig. 2.10. Rotación y Traslación de un vector. [7]

❖ Traslación

Para un sistema OUVW trasladado únicamente un vector p = pxi + pyj + pzk con respecto al sistema fijo OXYZ. La matriz homogénea será la matriz básica de traslación, determinada en la ecuación 2.1. [7]

$$T\ p\ = \begin{matrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{matrix} \tag{2.1}$$

Un vector cualquiera r, representado en OUVW por ruvw, tendrá como coordenadas en el sistema OXYZ, ecuación 2.2. [7]

$$\begin{matrix} r_x \\ r_y \\ r_z \\ 1 \end{matrix} = \begin{matrix} 1 & 0 & 0 & p_x & r_u \\ 0 & 1 & 0 & p_y & r_v \\ 0 & 0 & 1 & p_z & r_w \\ 0 & 0 & 0 & 1 & 1 \end{matrix} = \begin{matrix} r_u + p_x \\ r_v + p_y \\ r_w + p_z \\ 1 \end{matrix} \tag{2.2}$$

❖ Rotación

Supongamos que el sistema O'UVW sólo se encuentra rotado con respecto al sistema OXYZ. Las submatríz de rotación R3x3, ecuación 2.3, será la que defina la rotación. Se pueden definir tres matrices homogéneas básicas de rotación según el eje sobre el que se realice dicha rotación, ecuación 2.4-2.5-2.6. [7]

$${}^{i-1}R_i = \begin{matrix} cos\theta_i & -cos\alpha_i . sen\theta_i & sen\alpha_i . sen\theta_i \\ sen\theta_i & cos\alpha_i . cos\theta_i & -sen\alpha_i . cos\theta_i \\ 0 & sen\alpha_i & cos\alpha_i \end{matrix} \tag{2.3}$$

$$T\ x,\alpha\ = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sen\alpha & 0 \\ 0 & sen\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad (2.4)$$

$$T\ y,\varphi\ = \begin{matrix} cos\varphi & 0 & sen\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -sen\varphi & 0 & cos\varphi & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \quad (2.5)$$

$$T\ z,\theta\ = \begin{matrix} cos\theta & -sen\theta & 0 & 0 \\ sen\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad (2.6)$$

2.2.1.9. Parámetros Denavit-Hartenberg

El algoritmo de Denavit y Hartenberg fundamenta su utilidad en una forma específica para escoger los sistemas de coordenadas y determinar un conjunto de parámetros (parámetros DH). En principio, cada elemento de una cadena cinemática está asociado a dos articulaciones y pueden describirse mediante dos parámetros: **la distancia normal (a)** a los ejes de las articulaciones y **un ángulo de torsión (α)**.

Es importante notar que el ángulo α, no está en el plano de la vista frontal de la imagen (fig. 2.11.). Se ha intentado resaltar que la distancia **"a"**, es normal a ambos ejes, pero que los planos definidos por ella y cada eje son diferentes. [9]

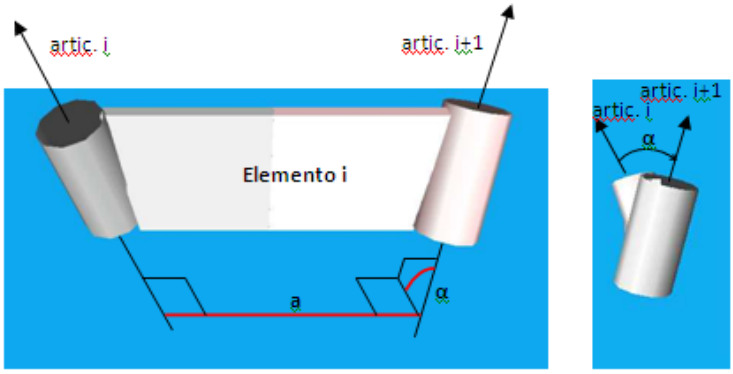


Fig. 2.11. Representación gráfica de los parámetros a y α. [9]

El eje de cada articulación comparte dos líneas normales (a) con ejes contiguos (fig. 2.12). La distancia entre ambas normales, en la dirección del eje de la articulación, se conoce como **d**. Finalmente, la distancia angular entre las dos normales, medida según el eje de la articulación se denomina θ.

En resumen, los parámetros DH son cuatro (4): θ, d, a y α. Los dos últimos (a y α) definen las características de cada eslabón o elemento de la cadena cinemática, mientras que los dos primeros (θ y d) determinan su relación con el elemento anterior. [9]

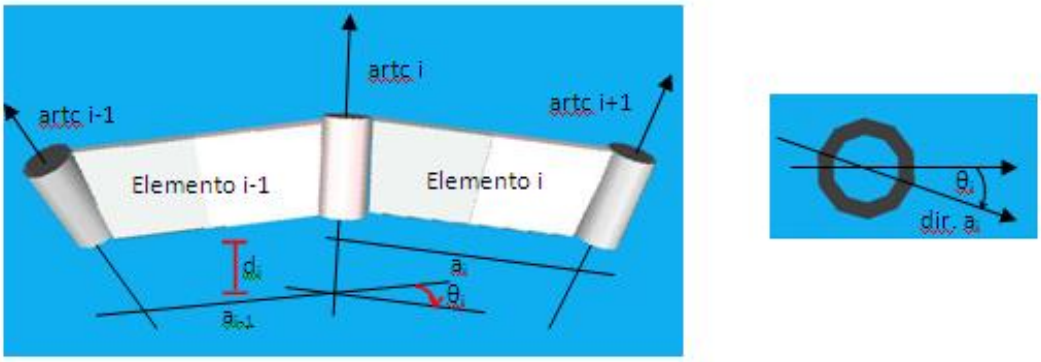


Fig. 2.12. Representación grafica de los parámetros θ y d. [9]

Tomando en cuenta lo expresado anteriormente se tiene:

- $a_{(i-1)}$

Definición: $a_{(i-1)}$ es el largo de la perpendicular entre los ejes de las articulaciones. Estos ejes corresponden a los ejes de giro, que corresponden a los ejes $Z_{(i-1)}$ y $Z_{(i)}$. Estos ejes deben considerarse rectas en el espacio. La recta perpendicular corresponde al segmento más corto entre las dos líneas axiales, y es perpendicular a ambas. [6]

Enfoque visual: $a_{(i-1)}$ puede visualizarse como un cilindro de eje z_{i-1} que se expande hasta tocar el eje i, entonces el radio del cilindro corresponde a *a(i-1)*. Si el eje es del tipo deslizante, entonces *a(i-1)* pasa a ser una variable y no un parámetro. [6]

- $\alpha_{(i-1)}$

Definición: Cantidad de rotación en torno a la perpendicular común de manera tal que los ejes de las articulaciones sean paralelos.

i.e. cuanto hay que rotar en torno al eje $X_{(i-1)}$ tal que $Z_{(i-1)}$ apunte en la misma dirección que Z_i. Rotación positiva siguiendo la regla de la mano derecha. [6]

- $d_{(i-1)}$

Definición: El desplazamiento a lo largo del eje Z_i necesario para conectar la perpendicular común $a_{(i-1)}$ con la perpendicular común a_i .

i.e. desplazamiento a lo largo de Z_i para conectar los ejes $X_{(i-1)}$ y X_i. [6]

- θ_i

Definición: Cantidad de rotación en torno al eje Z_i necesaria para alinear el eje $X_{(i-1)}$ con el eje X_i . [6]

2.2.1.10. Matriz de Paso Homogénea

Generalmente un robot de n grados de libertad se compone por n eslabones unidos por n articulaciones. A cada eslabón se le puede atribuir un sistema de referencia utilizando las transformaciones e innovaciones homogéneas, es posible entonces representar las rotaciones y traslaciones relativas entre los distintos eslabones que forman al robot. La matriz de transformación homogénea que representa la posición y orientación relativa entre los diferentes sistemas asociados a dos eslabones consecutivos del robot se designan i−1Ai. La matriz 0Ak, resultante del producto de las matrices i−1Ai con i desde 1 hasta k, es la que representa la cadena cinemática que forma el robot. Cuando se consideran todos los grados de libertad, a la matriz 0An se le llama T, matriz de transformación homogénea. [8]

Para representar la relación que existe entre dos sistemas de referencia asociados a eslabones, se utiliza la representación Denavit – Hartenberg (D-H) Este método matricial permite establecer de manera metódica un sistema de coordenadas ligado a cada eslabón i de una cadena articulada. También permite pasar de un sistema de coordenadas a otro mediante 4 transformaciones básicas que dependen de las características geométricas del eslabón. Estas transformaciones consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i con el sistema del elemento i − 1. [8]

. Dichas transformaciones serían:

- La rotación alrededor del eje z_{i-1} un ángulo θ_i.
- La traslación a lo largo de z_{i-1} una distancia d_i.
- La traslación a lo largo del eje x_i una distancia a_i.
- La rotación alrededor del eje x_i un ángulo α_i.

Teniendo estos valores que serían los parámetros D-H del eslabón i, la matriz de transformación que relaciona los sistemas de referencia {Si−1} y {Si} sería la dada en la ecuación 2.7. [8]

$$^{i-1}A_i = T\ z,\theta_i\ .(T\ 0{,}0,d_i\ .T\ a_i,0{,}0\ .T(x,\alpha_i) \tag{2.7}$$

Desarrollando esta ecuación, se obtiene la ecuación 2.8.

$$^{i-1}A_i = \begin{matrix} cos\theta_i & -cos\alpha_i.sen\theta_i & sen\alpha_i.sen\theta_i & a_i.cos\theta_i \\ sen\theta_i & cos\alpha_i.cos\theta_i & -sen\alpha_i.cos\theta_i & a_i.sen\theta_i \\ 0 & sen\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{2.8}$$

2.2.1.11. Selección de ejes

El origen del sistema de referencia del elemento 'i' se ubica en la articulación de su extremo (articulación i + 1), específicamente en la intersección de la normal 'ai' y el eje de la articulación final

El eje Z del elemento 'i' se escoge en coincidencia con el eje de la articulación final (i + 1), mientras que el eje X estará en la dirección de la normal 'ai'. El eje Yi cumple el producto Zi = Xi x Yi. [9]

La relación de los parámetros DH y los subíndices de los ejes puede parecer confusa. Para evitar este tipo de confusiones se debe recordar lo siguiente:

- Los ejes de un elemento se escogen en su extremo, no en su inicio (fig. 2.13.). [9]

- Los parámetros a y α, describen el elemento; y por tanto están directamente relacionados con su sistema de ejes, valga decir “sistema móvil”. [9]
- Los parámetros d y θ, describen la relación del elemento presente con el anterior; y por tanto son observados en la articulación de inicio y en relación con los ejes precedentes (con subíndice ‘i-1’). [9]

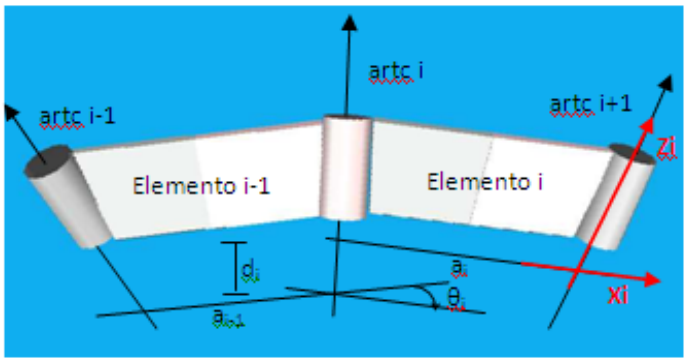


Fig. 2.13. Ubicación de los ejes del sistema S1. [9]

2.2.1.12. Convenciones de Denavit-Hartenberg

Denavit y R. Hartenberg hicieron masque definir los parámetros que llevan su nombre, también establecieron un conjunto de convenciones sobre los sistemas de ejes asociados que permiten sistematizar el análisis de robot manipuladores. A continuación se listan y comentan estas convenciones:

1.- Los eslabones se enumeran crecientemente, desde la base final, asignando el número 1 al eslabón que se encuentre unido a la base. Esta ultima (la base) recibirá el numero 0.

2.- Las articulaciones se enumeran también en forma creciente, comenzando por 1, desde la que une a la base con el primer elemento hasta la que sirve de junta al elemento terminal, como se muestra en la fig. 2.14.

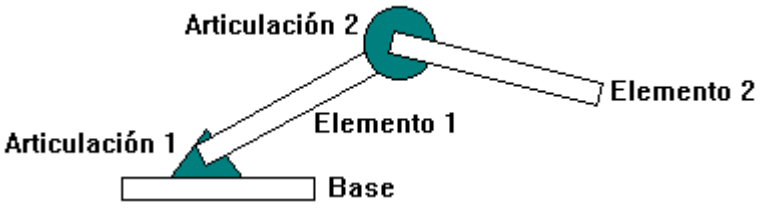


Fig. 2.14. Ejemplo de enumeración de los elementos y las articulaciones. [9]

3.- Localizar los ejes de las articulaciones: en las rotoides el eje será el eje de giro; y en las prismáticas, el eje del desplazamiento (fig. 2.15).

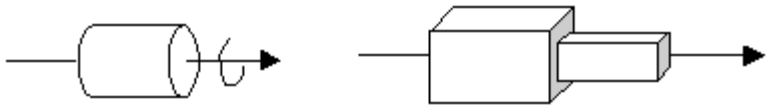


Fig. 2.15. Ejemplo de la ubicación de los ejes de las articulaciones rotoides y prismática. [10]

4.- Definir los ejes zi empezando desde la base, cada zi estará ubicado en el eje de la articulación correspondiente al extremo final del elemento (articulación i+1).

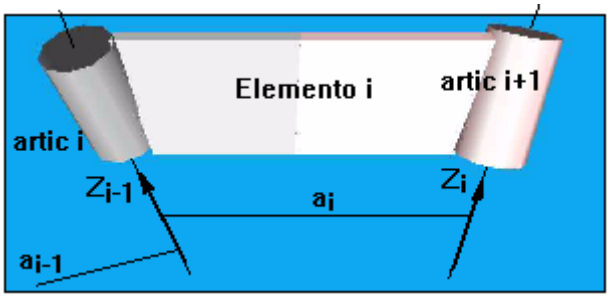


Fig. 2.16. Ejemplo de ubicación del eje Z. [10]

5.- El origen del sistema de coordenadas de la base (So) se escogerá arbitrariamente, como cualquier punto del eje Z0. También será arbitrario el sentido de X0, pero Y0 quedara determinado por las reglas del producto vectorial, tal que Z=XxY.

6.- El origen de cada sistema 'Si' quedara determinado por la intersección de la normal (a) común a cada eje Zi sistema dextrógiro (regla de la mano derecha).

7.- El sistema 'Sn' se ubicara en el elemento Terminal del robot y ele Zn conservara la dirección de Zn-1.

8.- θi será el ángulo en que debe girarse al eje Xi-1, alrededor de Zi-1, para quedar paralelo con Xi.

9.- di será la distancia que deberá desplazar a Xi-1, en la dirección de Zi-1, para quedar alineado con Xi.

10.- ai será la distancia que deberá desplazarse al Si-i, en la dirección de xi, para que su origen coincida con el de si.

11.- αi será el ángulo en que debe girarse a zi-1, alrededor de xi, para que alineado con zi.

Sobre estos cuatro puntos cabe observar que se trata de una redefinición de los parámetros descritos anteriormente. Lo mas destacable esa que quedan determinados los sentidos de los parámetros q y a, siguiendo la regla de la mano derecha. Así, si se alinean los cuatro dedos de la mano derecha con xi-1 y se los hace girar hacia xi, el pulgar nos dirá el signo de qi: positivo si queda apuntando en el sentido positivo de zi-1 y negativo en caso contrario (fig. 2.17). [9]

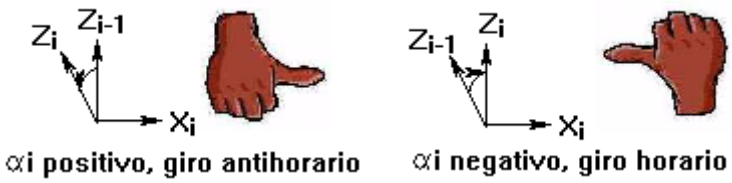


Fig. 2.17. Regla de la mano derecha. [10]

Lo propio es observable para el caso de ai: alineando los cuatro dedos de la mano derecha con zi-1 y girándolos hacia zi, debe observase si el pulgar apunta en el sentido positivo de xi (signo positivo) o en sentido contrario (signo negativo). [9]

12.- Deben obtenerse tantas matrices de paso homogéneas (${}^{i-1}A_i$) como elemento existente en la cadena cinemática, incluyendo la matriz ${}^{0}A_1$, correspondiente a la base.

13.- La matriz de transformación del robot se obtendrá como el producto de todas las matrices de paso homogéneas, mostradas en la ecuación 2.9.

$$T = {}^{0}A_1.\ {}^{1}A_2.\ {}^{2}A_3.\ {}^{3}A_4.\ldots.{}^{n-1}A_n = {}^{0}A_n \quad (2.9)$$

Dependiendo del tipo de articulación utilizada, cada matriz de paso homogénea tendrá como variable al ángulo q (rotoide) o la distancia d (prismática), a estos se les llama parámetro de la articulación (q) y es por su variación consciente que se logra controlar el posicionamiento del efector final. Una sencilla booleana, como se observa en la ecuación 2.10, discrimina si el parámetro de la articulación es un ángulo o una distancia , mediante una variable lógica llamada parámetro de vereshagin (s), la cual vale 0 si la articulaciones es rotoide o 1 si es prismática. [9]

$$q = sq + sd \quad (2.10)$$

En resumen, la solución del problema cinemática directo es la matriz de transformación del robot (t), la cual se encuentra mediante las matrices de paso homogéneo determinado sobre la base de los parámetros DH de cada eslabón. Con estas herramientas y el control de los parámetros de las articulaciones se logra posicionar el órgano o elemento terminal donde se decida. [9]

2.2.1.13. Centros de Masa y Centros de Gravedad

- **Centro de masa**

El Centro de masa es el punto en el cual se puede considerar concentrada toda la masa de un objeto o de un sistema. El Centro de masa de un objeto puede quedar fuera del cuerpo del objeto. El centro de masa de un anillo homogéneo está en su centro. [11]

Para efectos de este estudio, se toma en cuenta que las coordenadas de los centros de masa de cada elemento están referidas al sistema no inercial de los mismos, como se muestra en la fig. 2.18. Si el centro de masa es igual a cero, quiere decir que está localizado en el origen del sistema de coordenada que le corresponda.

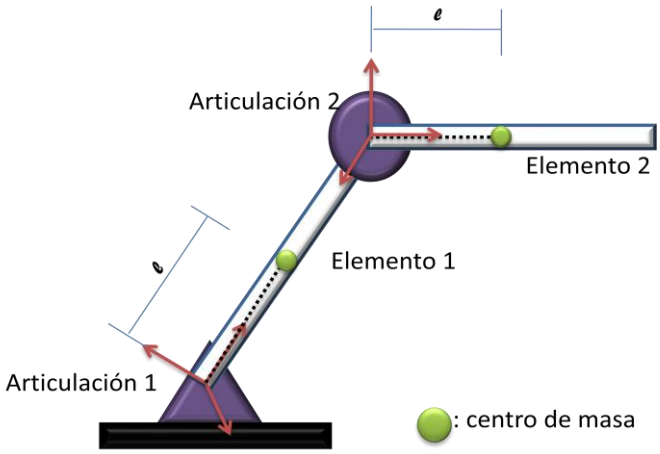


Fig. 2.18. Centro de masa de cada elemento para un robot de dos articulaciones.

Si el robot sostiene en su efector final una carga, ésta se debe tomar en cuenta para el cálculo del centro de masa del elemento unido a la herramienta terminal (fig. 2.19), y para ello se utiliza la ecuación 2.11, como se muestra a continuación:

$$X_{cm} = \frac{x_1.m_1+x_2.m_2}{m_1+m_2} \qquad (2.11)$$

Donde:

X_{cm}: coordenada x del nuevo centro de masa del eslabón n.

x_1: coordenada x del centro de masa del elemento n, referido al sistema de coordenadas de ese eslabón.

m_1: masa del último elemento.

x_2: coordenada x del centro de masa de la carga sostenida en el efector final, referido al sistema de coordenadas del eslabón n.

m_2: masa de la carga.

Otras coordenadas del centro de masa para sistemas de partículas se definen en forma similar. Para una distribución bidimensional de masas, las coordenadas son (X_{cm}; Y_{cm}). [11]

- **Centro de Gravedad**

La fuerza más corriente que actúa sobre un cuerpo es su propio peso. En todo cuerpo por irregular que sea, existe un punto tal en el que puedo considerarse en él concentrado todo su peso, este punto es considerado el centro de gravedad. El centro de gravedad puede ser un punto exterior o interior del cuerpo que se considere.

El conocimiento de la posición de los centros de gravedad, es de suma importancia en la resolución de problemas de equilibrio, porque son los puntos de aplicación de los vectores representativos de los respectivos pesos. [11]

En este trabajo el centro de gravedad está referido al sistema de coordenadas de la base del robot, como se observa en la fig. 2.19.

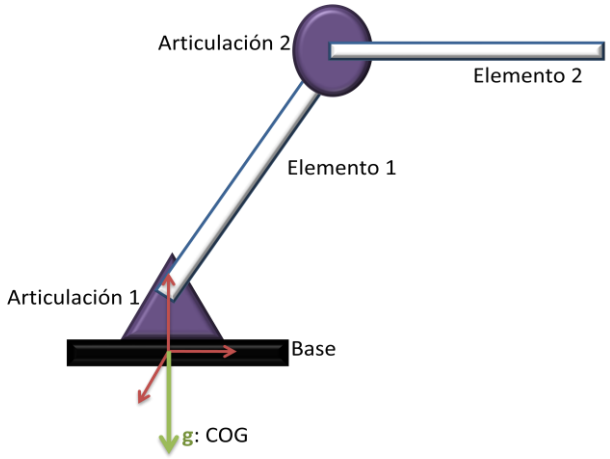


Fig. 2.19. Centro de Gravedad de un manipulador industrial.

En conclusión el centro de gravedad es el punto en el que se encuentran aplicadas las fuerzas gravitatorias de un objeto, o es decir es el punto en el que actúa el peso.

El equilibrio de una partícula o de un cuerpo rígido también se puede describir como estable o inestable en un campo gravitacional. Para los cuerpos rígidos, las categorías del equilibrio se pueden analizar de manera conveniente en términos del centro de gravedad. El Centro de gravedad es el punto en el cual se puede considerar que todo el peso de un cuerpo está concentrado y representado como una partícula. Cuando la aceleración debida a la gravedad sea constante, el

centro de gravedad y el centro de masa coinciden. [11]

Si la aceleración debida a la gravedad no es constante, el centro de masa y el centro de gravedad no coinciden. [11]

No olvide que la posición del centro de masa o centro de gravedad de un objeto depende de la distribución de la masa. Por lo tanto, para un objeto flexible como es el cuerpo humano, la posición del centro de gravedad cambia a medida que el objeto cambia su configuración (distribución de masa). [11]

2.2.1.14. Dinámica del robot

Dinámica es la parte de la física que describe la evolución en el tiempo de un sistema físico en relación a las causas que provocan los cambios de estado físico y/o estado de movimiento. El objetivo de la dinámica es describir los factores capaces de producir alteraciones de un sistema físico, cuantificarlos y plantear ecuaciones de movimiento o ecuaciones de evolución para dicho sistema de operación. También se puede definir dinámica como la parte de la mecánica que se ocupa del estudio del movimiento de los cuerpos sometidos a la acción de las fuerzas. [12]

La dinámica se ocupa de la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. [3]

2.2.1.15. Modelo Dinámico

El modelo dinámico describe los aspectos de un sistema que cambian con el tiempo. El modelo dinámico se utiliza para especificar e implementar los aspectos de control del sistema. Los modelos dinámicos contienen diagramas de estado, los cuales no son más que grafos cuyos nodos son estados y cuyos arcos son transiciones entre estados causadas por sucesos. [13]

El denominado modelo dinámico, tiene por objetivo conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo (fig. 2.20), además relaciona matemáticamente:

- La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración. [3]
- Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot). [3]
- Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.[3]

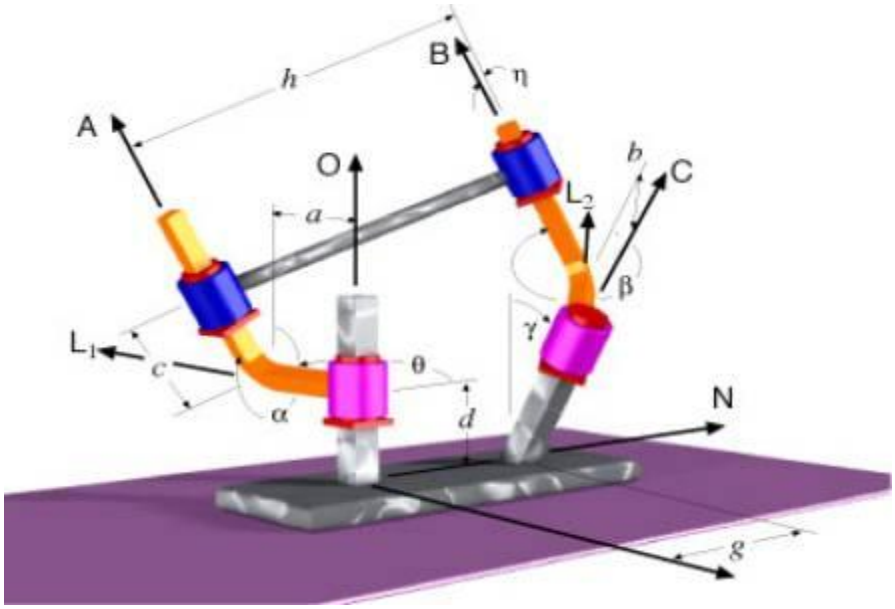


Fig. 2.20. Relación entre el movimiento del robot y las fuerzas implicadas en el mismo. [14]

❖ **Complejidad del modelo**

La obtención de este modelo para mecanismos de uno o dos grados de libertad no es excesivamente compleja, pero a medida que el número de grados aumenta, el planteamiento y la obtención del modelo dinámico se complica enormemente. Por este motivo no siempre es posible obtener un modelo dinámico expresado de una forma cerrada, esto es, mediante una serie de ecuaciones, normalmente de tipo diferencial de 2do orden, cuya integración permita conocer qué movimiento surge al aplicar unas fuerzas o qué fuerzas hay que aplicar para obtener un movimiento determinado. [3]

En general la complejidad del modelo dinámico de un robot [15]:

1. Crece con el número de grados de libertad del robot.
2. Interactividad entre movimientos (fuerzas de coriolis).
3. No siempre es posible su obtención en forma cerrada (ecuaciones diferenciales de orden 2 acopladas a integrar).
4. En ocasiones se debe recurrir a procedimientos numéricos iterativos.
5. Frecuentemente se realizan simplificaciones.
6. Necesidad de incluir los actuadores y su dinámica.
7. Caso especial: robots flexibles.

❖ Utilidad del modelo

El problema de la obtención del modelo dinámico de un robot es, por lo tanto, uno de los aspectos más complejos de la robótica, lo que ha llevado a ser obviado en muchas ocasiones. Sin embargo, el modelo dinámico es imprescindible para obtener los siguientes fines [3]:

- Simulación del movimiento del robot.
- Diseño y evaluación de la estructura mecánica del robot.
- Dimensionamiento de los actuadores.
- Diseño y evaluación de control dinámico del robot.

❖ Modelo Dinámico de la estructura de un robot

La obtención del modelo dinámico de un mecanismo, y en particular de un robot, se basa fundamentalmente en el planteamiento del equilibrio de fuerzas establecido en la segunda ley de Newton (ecuación 2.12), o su equivalente para movimientos de rotación, la denominada ley de Euler (ecuación 2.13): [3]

$$F = mv \tag{2.12}$$

$$\tau = Iw + w \times (Iw) \tag{2.13}$$

Así, en el caso simple de un robot monoarticular como el representado en la fig. 2.21, el equilibrio de fuerzas-pares daría como resultado la ecuación 2.14.

$$\tau = I\frac{d^2\theta}{dt^2} + MgLcos\theta = ML^2\theta + MgLcos\theta \qquad (2.14)$$

En donde se ha supuesto que toda la masa se encuentra concentrada en el centro de gravedad del elemento, que no existe rozamiento alguno y que no se manipula ninguna carga.

Para un par motor determinado, la integración de la ecuación 2.14 daría lugar a la expresión θ(t) y de sus derivadas dθ(t) y d²θ(t), con lo que sería posible conocer la evolución de la coordenada articular del robot y de su velocidad y aceleración. [3]

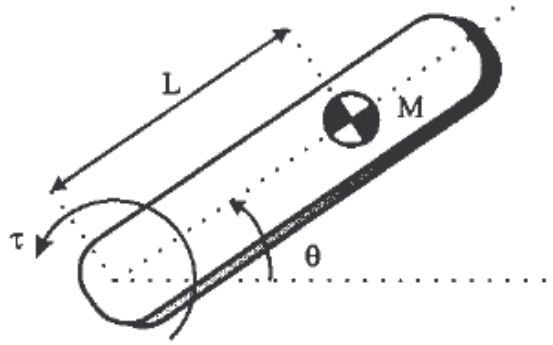


Fig. 2.21. Modelo de eslabón con masa concentrada. [3]

De forma inversa, si se pretende que θ(t) evolucione según una determinada función del tiempo, sustituyendo en la ecuación anterior, podría obtenerse el par τ(t) que sería necesario aplicar. Si el robot tuviese que ejercer alguna fuerza en su extremo, ya sea al manipular una carga o por ejemplo, realizar un proceso sobre alguna pieza, bastaría con incluir esta condición en la mencionada ecuación y proceder del mismo modo.

Se tiene así que del planteamiento del equilibrio de fuerzas y pares que intervienen sobre el robot se obtienen los denominados modelos dinámicos directo e inverso. [14]

❖ **Modelo dinámico directo e inverso**

Es importante tener bien claro que en l adinámica de manipuladores existen dos problemas básicamente, el modelo dinámico directo y el inverso (fig. 2.22).

- **Modelo dinámico directo**: expresa la evolución temporal de las coordenadas articulares del robot en función de las fuerzas y pares que intervienen. [14]
- **Modelo dinámico inverso**: expresa las fuerzas y pares que intervienen en función de la evolución de las coordenadas articulares y sus derivadas. [14]

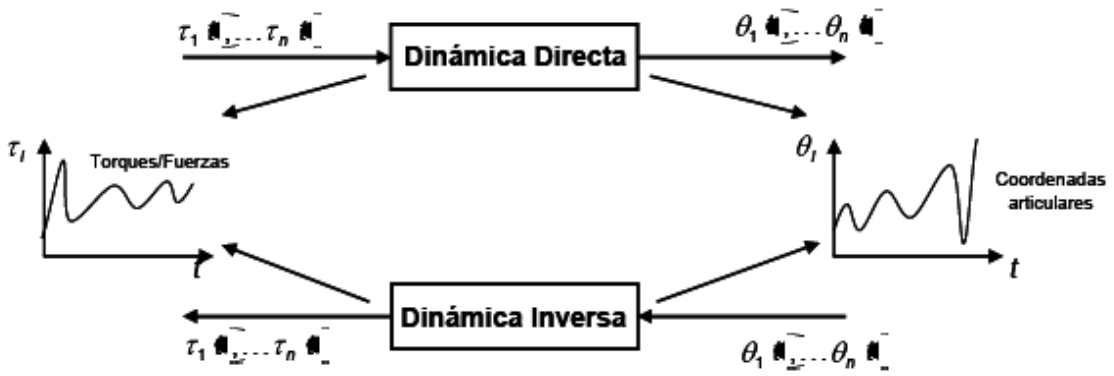


Fig. 2.22. Diagrama de relación entre Dinámica Directa e Inversa. [16]

La obtención del modelo dinámico de un robot ha sido y es objeto de estudio e investigación. Numerosos investigadores han desarrollado formulaciones alternativas, basadas fundamentalmente en la mecánica Newtoniana y Lagrangiana, con el objeto de obtener modelos manejables por los sistemas de cálculo de una manera más eficiente. [14]

2.2.1.16. Modelo dinámico de un robot mediante la formulación de Lagrange-Euler

Uicker en 1965, utilizo la representación de D-H basada en las matrices de transformación homogénea para formular el modelo dinámico de un robot mediante la ecuación de Lagrange. Este planteamiento utiliza, por tanto, las matrices i-1Ai que relacionan el sistema de coordenadas de referencia del elemento i con el elemento i-1. Se realizan en este caso operaciones de producto y suma innecesarias. Se trata de un procedimiento ineficiente desde el punto de vista computacional.

Puede comprobarse que el algoritmo es de un orden de complejidad computacional $O(n^{22})$, es decir, el número de operaciones a realizar crece con la potencia 4 del número de grados de libertad. Sin embargo, conduce a unas ecuaciones finales bien estructuradas donde aparecen de manera clara los diversos pares y fuerzas que intervienen en el movimiento.

Se presenta a continuación al algoritmo a seguir para obtener el modelo dinámico del robot por el procedimiento de Lagrange-Euler (L-E). [13]

- **Algoritmo computacional para el modelado dinámico por Lagrange-Euler.**

Los pasos a seguir para el desarrollo computacional del algoritmo es el siguiente: [3]

L-E 1. Asignar a cada eslabón un sistema de referencia de acuerdo con las normas DH.

L-E 2. Obtener las matrices de transformación 0Ai para cada elemento i aplicando la ecuación 2.9.

L-E 3. Obtener las matrices uij definidas por:

$$U_{ij} = \frac{d\,{}^{0}A_i}{dq_j} \tag{2.15}$$

L-E 4. Obtener las matrices uijk definidas por:

$$U_{ijk} = \frac{d^2\ {}^0A_i}{dq_j^2} \tag{2.16}$$

L-E 5. Obtener las matrices de pseudoinercias Ji para cada elemento, que vienen definidas por:

$$J_i = \begin{matrix} x_i^2 dm & x_i\ y_i dm & x_i\ z_i dm & x_i\ dm \\ y_i\ x_i dm & y_i^2 dm & y_i\ z_i dm & y_i dm \\ z_i\ x_i dm & z_i\ y_i dm & z_i^2 dm & z_i dm \\ x_i\ dm & y_i dm & z_i dm & dm \end{matrix} \tag{2.17}$$

Donde las integrales están extendidas al elemento i considerado, y (*xi yi zi*) son las coordenadas del diferencial de masa dm respecto al sistema de coordenadas del elemento.

L-E 6. Obtener la matriz de inercia **D**=[dij] cuyos elementos vienen definidos por:

$$D = d_{ij} = \sum_{k=\max\ i,j}^{n} Traza\ U_{kj} J_k U_{ki}^T \tag{2.18}$$

con i, j= 1,2,…,n

n: número de grados de libertad

L-E 7. Obtener los términos hikm definidos por:

$$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} Traza(U_{jkm} J_j U_{ji}^T) \tag{2.19}$$

con i, k, m= 1, 2, …, n

L-E 8. Obtener la matriz columna de fuerzas de Coriolis y centrípeta **H**=$[h_i]^T$ cuyos elementos vienen definidos por:

$$h_i = \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm} q_k q_m \tag{2.20}$$

L-E 9. Obtener la matriz columna de fuerzas de gravedad **C**=$[c_i]^T$ cuyos elementos están definidos por:

$$c_i = \sum_{j=1}^{n} -m_j g U_{ij}\ {}^{j}r_j \qquad (2.21)$$

con i=1, 2, …, n

g: es el vector de fuerzas de gravedad expresado en el sistema de la base {S0} y viene expresado por (*gx0, gy0, gz0, 0*)

i**r**j: es el vector de coordenadas homogéneas del centro de masa del elemento j expresado en el sistema de referencia del elemento i.

L-E 10. La ecuación dinámica del sistema será:

$$\tau = D\ q\ .q + H\ q,q\ + C(q) \qquad (2.22)$$

donde τ es el vector de fuerzas y pares motores efectivos aplicados sobre cada coordenada qi.

2.2.1.17. Modelo dinámico de un robot mediante la formulación de Newton-Euler

La obtención del modelo dinámico de un robot a partir de la formulación Langragiana conduce a un algoritmo con un coste computacional de orden O(n4). Es decir, el número de operaciones a realizar crece con la potencia cuarta número de grados de libertad. En el caso habitual de robots de 6 grados de libertad, este de número de operaciones hace al algoritmo presentado en el epígrafe anterior materialmente inutilizable para ser utilizado en tiempo real.

La formulación de Newton-Euler parte del equilibrio de pares y fuerzas presentados en las ecuaciones 2.12 y 2.13.

Un adecuado desarrollo de estas ecuaciones conduce a una formulación recursiva en la que se obtienen la posición, velocidad y aceleración del eslabón i referidos ala base del robot a partir de los correspondientes del eslabón i-1 y del

movimiento relativpo0 de la articulación i. De este modo, partiendo del eslabón 1 se llega al eslabón n. Con estos datos se procede a obtener las fuerzas y pares actuantes sobre el eslabón i referidos a la base del robot a partir de los correspondientes al eslabón i-1, recorriéndose de esta forma todos los eslabones desde el eslabón hasta el 1.

El algoritmo se basa en operaciones vectoriales (con productos escalares y vectoriales entre magnitudes vectoriales, y productos de matrices de vectores) siendo mas eficiente en comparación con las opresiones matriciales asociadas a la formulación Langragiana. De hecho, el orden de complejidad computacional de la formulación recursiva de Newton-Euler es O(n) lo que indica que depende directamente del número de grados de libertad.

El algoritmo se desarrolla en los siguientes pasos.

- **Algoritmo computacional para el modelado dinámico por Newton-Euler**

N-E 1. Asignar a cada eslabón un sistema de referencia de acuerdo con las normas de D-H.

N-E 2. Obtener las matrices de rotación (ver ecuación 2.3) ${}^{i-1}R_i$ y sus inversas ${}^{i}R_{i-1} = ({}^{i-1}R_i)^{-1} = ({}^{i-1}R_i)^T$.

N-E 3. Establecer las condiciones iniciales.

Para el sistema de la base:

${}^0w_0: velocidad\ angular = [0,0,0]^T$

${}^0w_0: aceleración\ angular = [0,0,0]^T$

${}^0v_0: velocidad\ lineal = [0,0,0]^T$

${}^0v_0: aceleración\ lineal = [g_x, g_y, g_z]^T$

La velocidad y aceleración angular y la velocidad lineal son típicamente nulas salvo que la base del robot esté en movimiento. Para el extremo del robot se conocerá la fuerza y el par ejercido externamente ${}^{n+1}f_{n+1}$ y ${}^{n+1}n_{n+1}$

Z_0: $vector\ unitario\ en\ dirección\ al\ eje\ de\ la\ articulación = [0,0,1]^T$

${}^iP_i$: $coordenadaas\ del\ origen\ del\ sistema\ \ S_i\ \ respecto\ a\ \ S_{i-1}\ \ = [a_i, d_i sen\alpha_i, d_i cos\alpha_i]$

$$ {}^iP_i = [a_i, d_i sen\alpha_i, d_i cos\alpha_i] \tag{2.23}$$

${}^is_i$: $coordenadas\ del\ centro\ de\ masas\ del\ eslabón\ i respecto\ del\ \ sistema\ \ S_i$

${}^iI_i$: , $matriz\ de\ inercia\ \ del\ eslabón\ i\ respecto\ de\ su\ centro\ de\ masas\ expresado\ en\ \{S_i\}$

Para i= 1…n realizar los pasos 4 a 7:

N-E 4. Obtener la velocidad angular del sistema Si

$$ {}^iw_i = \begin{matrix} {}^iR_{i-1}({}^{i-1}w_{i-1} + z_0 q_\iota) & si\ el\ eslabón\ i\ es\ de\ rotación \\ {}^iR_{i-1}\,{}^{i-1}w_{i-1} & si\ el\ eslabón\ i\ es\ de\ traslación \end{matrix} \tag{2.24}$$

N-E 5. Obtener la aceleración angular del sistema Si

$$ {}^iw_i = \begin{matrix} {}^iR_{i-1}({}^{i-1}w_{i-1} + z_0 q_\iota) + {}^{i-1}w_{i-1} \times z_0 q_i & si\ el\ eslabón\ i\ es\ de\ rotación \\ {}^iR_{i-1}\,{}^{i-1}w_{i-1} & si\ el\ eslabón\ i\ es\ de\ traslación \end{matrix} \tag{2.25}$$

N-E 6. Obtener la aceleración lineal del sistema i:

$$ {}^iv_i = \begin{matrix} {}^iw_i \times {}^ip_i \times {}^iw_i \ \ {}^iw_i \times {}^ip_i \ + \ {}^iR_{i-1}\,{}^{i-1}v_{i-1} & si\ el\ eslabón\ i\ es\ de\ rotación \\ {}^iR_{i-1}(z_0 q_\iota + {}^{i-1}v_{i-1}) + {}^iw_i \times {}^ip_i + & \\ +2\ {}^iw_i \times {}^iR_{i-1} z_0 q_i + {}^iw_i \ \ {}^iw_i \times {}^ip_i & si\ el\ eslabón\ es\ de\ traslación \end{matrix} \tag{2.26}$$

N-E 7. Obtener la aceleración lineal del centro de gravedad del eslabón i:

$$ {}^ia_i = {}^iw_i \times {}^iS_i + {}^iw_i \ \ {}^iw_i \times {}^ip_i \ + \ {}^iv_i \tag{2.27}$$

Para i=n… 1 realizar los pasos 8 a 10.

N-E 8. Obtener la fuerza ejercida sobre el eslabón i:

$$ {}^{i}f_i = {}^{i}R_{i+1}\,{}^{i+1}f_{i+1} + m_i\,{}^{i}a_i \tag{2.28} $$

N-E 9. Obtener el par ejercido por el eslabón i:

$$ {}^{i}n_i = {}^{i}R_{i+1}[{}^{i+1}n_i + ({}^{i+1}R_i\,{}^{i}p_i) \times {}^{i+1}f_{i+1}] + ({}^{i}p_i + {}^{i}s_i) \times m_i\,{}^{i}a_i + $$

$$ + {}^{i}I_i\,{}^{i}w_i + {}^{i}w_i \times ({}^{i}I_i\,{}^{i}w_i) \tag{2.29} $$

N-E 10. Obtener la fuerza o par aplicado a la articulación i:

$$ \tau_i = \begin{matrix} {}^{i}n_i^T\,{}^{i}R_{i+1}z_0 \\ {}^{i}f_i^T\,{}^{i}R_{i+1}z_0 \end{matrix} \tag{2.30} $$

Donde τ es el par o fuerza efectivo (par motor menos pares de rozamiento o perturbación). [3]

2.2.2. MATLAB

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyo especializado, denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el

'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc. Es un entorno de cálculo técnico, que se ha convertido en estándar de la industria, con capacidades no superadas en computación y visualización numérica.

De forma coherente y sin ningún tipo de fisuras, integra los requisitos claves de un sistema de computación técnico: cálculo numérico, gráficos, herramientas para aplicaciones específicas y capacidad de ejecución en múltiples plataformas. Esta familia de productos proporciona al estudiante un medio de carácter único, para resolver los problemas más complejos y difíciles. [17]

- **Scripts y funciones**

Un script es una colección de comandos Matlab escritos en un archivo-m (archivos con extensión .m) y que pueden ser ejecutados todos de una vez tan sólo escribiendo el nombre del archivo (sin la extensión .m). Esto es útil cuando se tiene un procedimiento que se debe aplicar repetitivamente y no se quiere estar escribiendo cada vez todos estos comandos, en su lugar se escriben en un archivo y luego se llama a través del nombre de este archivo.

Para poder ejecutar estos archivos-m es necesario localizar este archivo en los directorios en donde Matlab buscará por ellos. El conjunto de estos directorios se puede ver con el comando path dentro de Matlab (ejecutar “help path” para ver información de como agregar otros directorios para búsquedas).

Dentro de estos archivos-m se pueden también definir funciones. Una función es un conjunto de líneas de comandos Matlab pero que poseen un espacio de variables aparte (esto quiere decir que dentro de una función no serán “visibles” las variables definidas fuera de ésta, por lo que si se quiere esto así sea es necesario de pasarle a la función tal argumento), además de retornar un valor específico al final de la función. [18]

- **Entorno de trabajo de Matlab**

El entorno de trabajo: A partir de la versión 5.0, el entorno de Matlab ha mejorado mucho, haciéndose mucho más gráfico e intuitivo. Los principales componentes de dicho entorno son el explorador de caminos de búsqueda (Path Browser), el editor y depurador de errores (Editor/Debugger) y el visualizador del espacio de trabajo (Workspace Browser). [19]

Path Browser: Matlab puede llamar a una gran variedad de funciones, tanto propias como programadas por los usuarios. A veces, puede haber funciones distintas que tengan el mismo nombre. Por tanto, es interesante saber cómo Matlab busca cualquier función que se le pida que ejecute. La clave es el camino de búsqueda (search path) que el programa utiliza cuando encuentra el nombre de una función. El search path es una lista de directorios que se puede ver y modificar mediante la orden path, o utilizando el Path Browser (Submenú Set Path en el menú File). [19]

El directorio actual: El concepto de directorio actual o de trabajo es crucial en Matlab. Es el directorio donde el usuario debe guardar los diferentes archivos que genere en las sesiones, para que Matlab pueda detectarlos. Puede consultar el directorio en que se encuentra con la orden pwd. El contenido de dicho directorio puede obtenerse con la orden dir. Para cambiar el directorio actual se utiliza la orden cd (Change Directory) seguido del nombre del nuevo directorio. Ejecutando cd .., se sube un nivel en la jerarquía de directorios. [19]

Editor/Debbuger: En Matlab tienen particular importancia los M-archivos, esto es, archivos con la extensión "*.m", los cuales son archivos de texto ASCII que contienen un cierto conjunto de órdenes de Matlab. La importancia de estos archivos es que al teclear su nombre en la línea de órdenes de Matlab y pulsar Return , se ejecutan todas las órdenes contenidas en dicho archivo. Matlab dispone de un editor propio que permite tanto crear y modificar estos archivos (proceso de edición-Editor), como ejecutarlos paso a paso para detectar errores (proceso de depuración-Debugger). [19]

Workspace Browser: El espacio de trabajo (Workspace) de Matlab es el conjunto de variables que en un determinado momento están definidas en la memoria del programa. Para obtener información sobre el workspace se pueden utilizar las órdenes who y whos. La segunda proporciona una información más detallada que la primera. [19]

- **Interfaz Gráfica de Usuario**

La **interfaz gráfica de usuario**, conocida también como **GUI** (del inglés *graphical user interface*) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Habitualmente las acciones se realizan mediante manipulación directa, para facilitar la interacción del usuario con la computadora. Surge como evolución de los intérpretes de comandos que se usaban para operar los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X, Aqua.

En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático. [20]

Creación de una GUI

Para iniciar nuestro proyecto, lo podemos hacer de dos maneras:

- Ejecutando la siguiente instrucción en la ventana de comandos:

  >> guide

- Haciendo un click en el ícono que muestra la fig. 2.23.

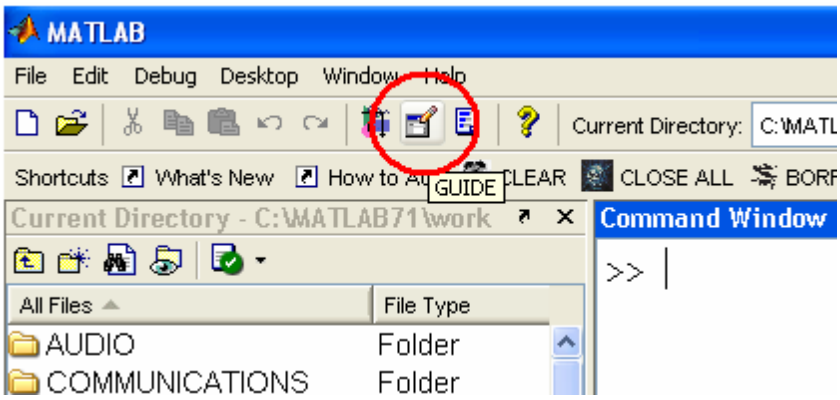


Fig. 2.23. Icono GUIDE. [21]

Se presenta el cuadro de diálogo mostrado en la fig. 2.24.

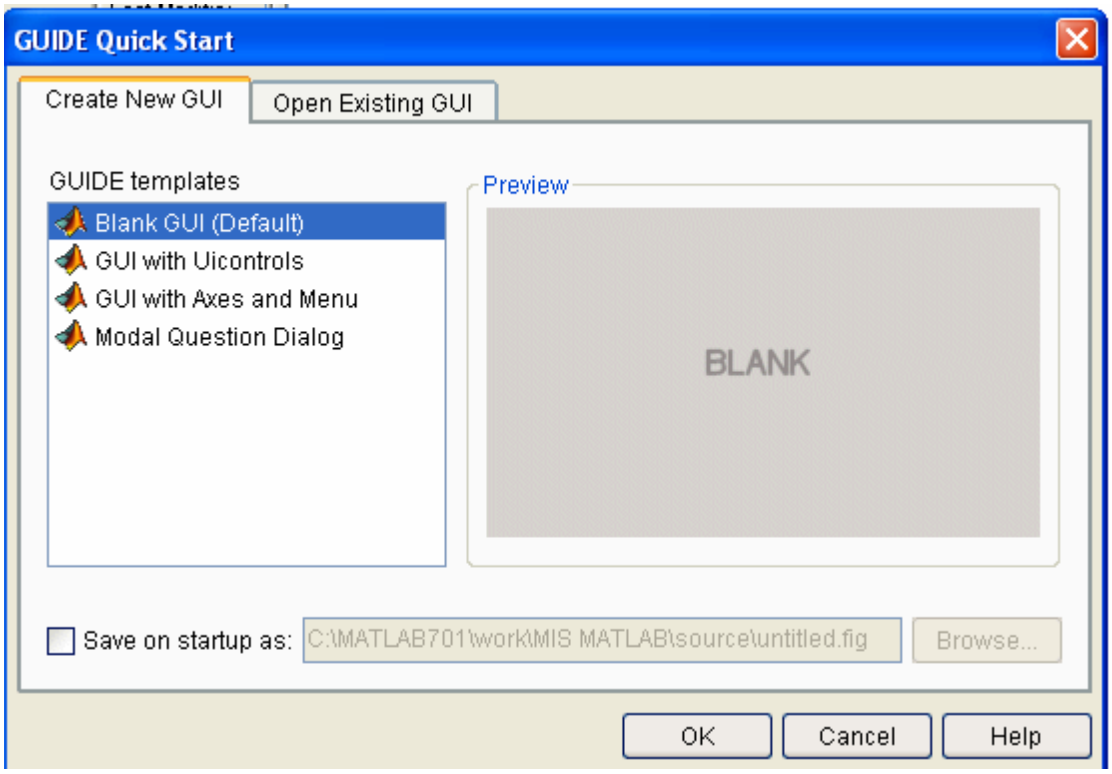


Fig. 2.24. Ventana de inicio de GUI. [21]

En la fig. 2.25 se presentan las siguientes opciones:

a) Blank GUI (Default)

La opción de interfaz gráfica de usuario en blanco (viene predeterminada), nos presenta un formulario nuevo, en el cual podemos diseñar nuestro programa.

b) GUI with Uicontrols

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) GUI with Axes and Menu

Esta opción es otro ejemplo el cual contiene el menú File con las opciones Open, Print y Close. En el formulario tiene un *Popup menu*, un *push button* y un objeto *Axes*, podemos ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo click en el botón de comando.

d) Modal Question Dialog

Con esta opción se muestra en la pantalla un cuadro de diálogo común, el cual consta de una pequeña imagen, una etiqueta y dos botones *Yes* y *No*, dependiendo del botón que se presione, el GUI retorna el texto seleccionado (la cadena de caracteres 'Yes' o 'No').

Se elige la primera opción, *Blank GUI,* y se observa la fig. 2.25.

Fig. 2.25. Entorno de diseño de GUI. [21]

La interfaz gráfica cuenta con las herramientas mostradas en la fig. 2.26.

	Alinear objetos.
	Editor de menú.
	Editor de orden de etiqueta.
	Editor del M-file.
	Propiedades de objetos.
	Navegador de objetos.
	Grabar y ejecutar (ctrl. + T).

Fig. 2.26. Herramientas GUI. [21]

Para obtener la etiqueta de cada elemento de la paleta de componentes se ejecuta: *File>>Preferentes* y seleccionamos *Show names in component palette.* Tenemos la presentación de la fig. 2.27.

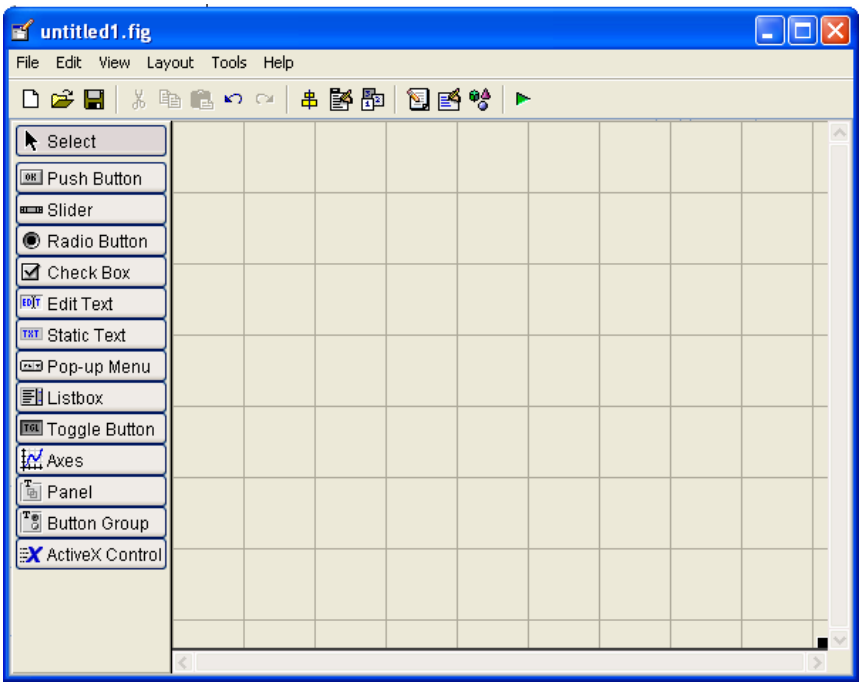


Fig. 2.27. Entorno de diseño: componentes etiquetados. [21]

La fig. 2.28 muestra una descripción de los componentes.

Control	*Valor de estilo*	*Descripción*
Check box	'checkbox'	Indica el estado de una opción o atributo
Editable Text	'edit'	Caja para editar texto
Pop-up menu	'popupmenu'	Provee una lista de opciones
List Box	'listbox'	Muestra una lista deslizable
Push Button	'pushbutton'	Invoca un evento inmediatamente
Radio Button	'radio'	Indica una opción que puede ser seleccionada
Toggle Button	'togglebutton'	Solo dos estados, "on" o "off"
Slider	'slider'	Usado para representar un rango de valores
Static Text	'text'	Muestra un string de texto en una caja
Panel button		Agrupa botones como un grupo
Button Group		Permite exclusividad de selección con los radio button

Fig. 2.28. Descripción de los componentes de una GUI. [21]

Para personalizar cada elemento se tiene la opción de Property Inspector, que se puede acceder a ésta haciendo doble click sobre el componente (fig. 2.29).

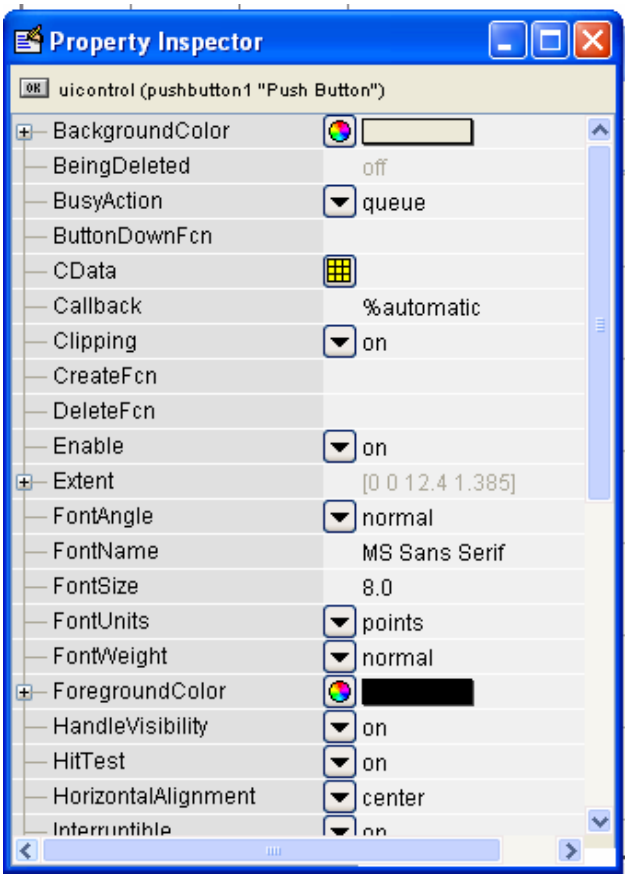


Fig. 2.29. Entorno Property Inspector. [21]

Permite ver y editar las propiedades de un objeto.

Al hacer click derecho en el elemento ubicado en el área de diseño, una de las opciones más importantes es *View Callbacks*, la cual, al ejecutarla, abre el archivo *.m* asociado a nuestro diseño y nos posiciona en la parte del programa que corresponde a la subrutina que se ejecutará cuando se realice una determinada acción sobre el elemento que estamos editando. [21]

Funcionamiento de una GUI

Una aplicación GUIDE consta de dos archivos: *.m* y *.fig*. El archivo *.m* es el que contiene el código con las correspondencias de los botones de control de la interfaz y el archivo *.fig* contiene los elementos gráficos. Cada vez que se adicione un nuevo elemento en la interfaz gráfica, se genera automáticamente código en el archivo *.m*. Para ejecutar una Interfaz Gráfica, si la hemos etiquetado con el nombre *curso.fig*, simplemente ejecutamos en la ventana de comandos >> *curso*. O haciendo click derecho en el m-file y seleccionando la opción *RUN*. [21]

CAPÍTULO III

3.1. METODOLOGÍA DE LA INVESTIGACIÓN

La modalidad de investigación utilizada en el presente trabajo de grado, es la de Proyecto Factible, por considerar, que según lo expresado en el manual de la UPEL-IPMAR (1998); ella permite: “Desarrollar modelos operativos para resolver problemas de la realidad.” [22]

Cabe destacar que el uso de esta modalidad implica, el apoyo en estudios de tipo documental o de campo; entendiendo el primero como aquella modalidad investigativa que se fundamenta en el uso de textos y documentos para dar explicación o respuesta teórica a la problemática y a los objetivos en estudio y a la modalidad de campo, como aquella que permite según el manual de la UPEL-IPMAR (1998) : “Desarrollar un análisis sistemático de problemas, a fin de descubrirlos, comprender sus causas y efectos, desde lo interpretativo, evaluativo y lo cuantitativo”. [22]

Tomando en cuenta la modalidad descrita anteriormente se afianza principalmente en otros tipos de investigación en el cual se toma como criterio el lugar y los recursos donde se obtiene la información requerida, la investigación es de tipo documental ya que, según Zorrilla (1993): “es aquella que se realiza a través de la consulta de documentos (libros, revistas, periódicos, memorias, anuarios, registros, códices, constituciones, etc.).” También este tipo de investigación es del tipo documental porque se realiza a través de la consulta de documentos, tesis y artículos.

Esta investigación también es del tipo aplicada, según Zorrilla (1993): “la investigación aplicada, guarda intima relación con la básica, pues depende de los descubrimientos y avances de la investigación básica y se enriquece con ellos, pero se caracteriza por su interés en la aplicación, utilización y consecuencias prácticas de los conocimientos. La investigación aplicada busca el conocer para hacer, para actuar, para construir, para modificar.” [23]

Este proyecto está basado en la creación de una Toolbox e interfaz grafica de usuario en Matlab para obtener el modelo dinámico de un robot, por lo tanto el tipo de investigación que se va utilizar es del tipo documental ya que se piensa desarrollar mediante estudios previos realizados con el fin de profundizar conocimientos en el área de la robótica. Además este proyecto poseerá un modelo de investigación del tipo especial ya que se realizará con la finalidad de crear una herramienta que facilite el posterior estudio en esta área.

3.2. NIVEL DE INVESTIGACIÓN

En cuanto al nivel de investigación se puede considerar intermedio debido a que para el desarrollo del trabajo necesita ciertos conocimientos básicos, pero se espera que el proceso sea de fácil usabilidad para los usuarios con el fin de que conozcan ciertas características para el análisis de los robots como son: la matriz de inercia, la matriz de fuerzas de gravedad, las matrices de transformación homogénea de sus elementos, entre otras que se encuentran dentro de los algoritmos computacionales y finalmente conseguir el modelo dinámico del robot deseado.

3.3. RECOLECCIÓN DE INFORMACIÓN

La recolección de la información está basada en investigaciones hechas con anterioridad las cuales brindan la ayuda necesaria para la compresión del problema planteado. Esta información permitirá sustentar conocimientos ya obtenidos así como colocar las bases teóricas fundamentales para dar inicio a la investigación.

La información recolectada se basó en:

- Archivos electrónicos (Descargas realizadas a través de Internet)
- Trabajos de grado, Tesis doctorales de otras Universidades.
- Trabajos de grados de la Universidad de Carabobo.
- Paginas de Internet Referentes al tema de robótica.

3.4. FASES DE LA INVESTIGACIÓN

Fase I: Recopilación de información

En esta primera etapa se realizó la investigación y estudio profundo del tema de los Robots Industriales y las metodologías Lagrange-Euler y Newton-Euler para la elaboración del modelo dinámico de los mismos. Luego de la recolección, análisis y síntesis de la información se ha llegado a la comprensión necesaria del estudio acerca de: Conceptos básicos de la robótica, Tipos de robots, Herramientas matemáticas de interés, Convenciones de Denavit-Hartenberg, dinámica directa, dinámica inversa y otros puntos de interés acerca del tema.

Fase II: Desarrollo de la Toolbox

Para la elaboración de esta fase fue necesario asimilar y repasar los conocimientos obtenidos durante la documentación de la tesis ya que a partir de aquí se obtuvieron las herramientas necesarias para conseguir los modelos dinámicos de los robots. Ya teniendo las ideas y realizada la toma de decisiones entre lo que definiríamos como scripts y funciones, se procedió al desarrollo de los mismos para la aplicación en MATLAB de los algoritmos correspondientes a las formulaciones de Langrange-Euler y Newton-Euler, con el objeto de lograr el diseño de una Toolbox que permita la modelación dinámica de los Robots Industriales.

Fase III: Evaluación de la Toolbox

Se evaluó la Toolbox en MATLAB para la modelación dinámica de manipuladores industriales con ejemplos de Robots Industriales con base en la información recopilada anteriormente, permitiendo verificar la efectividad de la misma y asegurar el correcto funcionamiento evitando así errores en la ejecución o llamados de los scripts y funciones.

Fase IV: Desarrollo de la interfaz grafica de usuario (GUI)

En esta fase se definieron las diferentes variables de interés para la construcción de las ventanas principales de la interfaz, entre éstas se encuentran: funcionalidad, confiabilidad, eficiencia, facilidad de uso, y otros puntos de interés a considerar. La GUI principal esta constituida por diferentes ventanas, las cuales cumplen una tarea especifica. La ventana de presentación *DibotMat* da paso a la siguiente interfaz *robotgui* para comenzar la modelación dinámica del robot, donde serán definidos por el usuario: el número de articulaciones, los parámetros de Denavit-Hartenberg, masa y centro de masa del los enlaces, la gravedad del robot en estudio referida a la base del mismo y si éste sostiene alguna carga en su efector final, la masa y centro de masa de ella. Una vez definidos dichos parámetros se va a la siguiente interfaz *modelogui*, donde se selecciona el algoritmo computacional deseado, inclusive se puede seleccionar los dos, o si se desea, generar alguna función de las utilizadas en los modelos.

Las ventanas cuentan con un menú en su parte superior donde se le suministra al usuario ayuda básica para el uso o exploración de las mismas y los fundamentos teóricos necesarios para adquirir el conocimiento básico sobre el tema en estudio.

Las ventanas que muestran resultados, se diseñaron con la finalidad de facilitar el uso, siguiendo una estructura similar para todas, los datos que se deben introducir van en los cuadros de marco azul y los resultados impresos se arrojarán en cuadros de bordes rojos para evitar confusiones o resaltar la diferencia entre ellos.

Fase V: Evaluación de la interfaz grafica de usuario

Para la evaluación de la Interfaz Gráfica de Usuario se ha procedido a explorar con detalle cada aspecto en su ejecución con la finalidad de que la misma cumpla cabalmente los objetivos trazados en este trabajo. Para hacer dicho análisis se estudiaron los mismos ejemplos tomados para la evaluación de la Toolbox (fase III) y así se logra comprobar su buen funcionamiento.

Fase VI: Comprobación de la Toolbox e Interfaz Gráfica de Usuario

Una vez realizada las evaluaciones de las fases anteriores, se ha comprobado que por medio del uso de la Toolbox y la Interfaz Gráfica de Usuario se ha llegado a los mismos resultados. Ahora bien, se necesita verificar si estos resultados son idóneos para la modelación dinámica de Robots Industriales y para ello se ha tomado como ayuda Planar Manipulator Toolbox (PLANMANT), elaborada por León Zlajpah. Debido a que este trabajo se basó en la dinámica inversa (dadas las coordenadas articulares generar las fuerzas o pares ejercidas por los brazos del robot), entonces la manera cómo se validó fue sencilla ya que se utilizó la dinámica directa (dado las fuerzas y pares actuantes en el robot generar las coordenadas de cada articulación del mismo) a través de una simulación en PLANMANT.

CAPÍTULO IV

Una vez estudiada las herramientas conceptuales de las formulaciones de Lagrange-Euler y Newton-Euler, además de la teoria de Robótica y Vision Industrial, se ha procedido a desarrollar los algoritmos computacionales correspondientes para el diseño de la toolbox, estableciendo para los elementos del robot masas concentradas en sus respectivos centros de gravedad.

Es importante resaltar, que para el desarrollo de esta toolbox se utiliza el entorno de programación de MATLAB v.7.6.0.324 (R2008a) bajo el sistema operativo Windows XP.

4.1. DESARROLLO COMPUTACIONAL DEL ALGORITMO DE LAGRANGE-EULER

En la fig. 4.1 se busca representar de manera gráfica y resumida cómo se ha desarrollado el algoritmo.

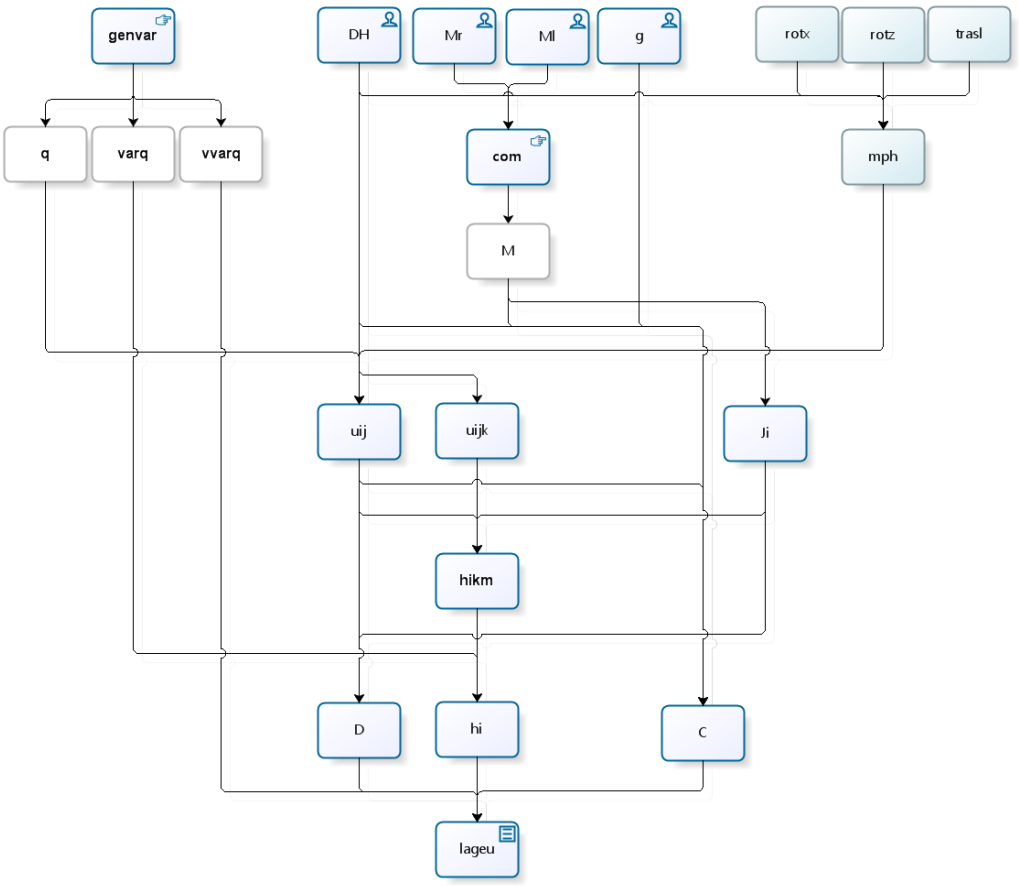


Fig. 4.1.- Algoritmo computacional de Langrage-Euler.

Donde:

 : Función que debe ser llamada manualmente en el comand windows por el usuario.

 : Variables generadas de la función llamada manualmente.

 : Parámetros que debe introducir el usuario en el comand windows.

 : Función que es utilizada sólo dentro de otra función.

 : Función utilizada en el script.

 : Script del algoritmo.

4.1.1. Función que debe ser llamada manualmente en el comand windows por el usuario.

4.1.1.1. FUNCIÓN ***genvar***

Esta función solo se debe llamar desde el command windows y la entrada es un vector de caracteres que indica de qué tipo es cada articulacion del robot, el usuario definirá para las rotoides el carácter ‘r’ y para las articulaciones prismáticas será ‘p’. Como salida se obtienen tres vectores simbólicos que son *q, varq* y *vvarq.*

4.1.1.2. FUNCIÓN ***com***

Esta función se desarrolló con la finalidad de que dado los centros de masa y la masa de los eslabones y la carga sostenida por el robot, genere como salida la matriz M de acuerdo a lo detallado en el capitulo II, por lo que se deduce que los parámetros de entrada de esta función son Mr y Ml. FUNCIÓN *com* sólo puede ser llamada desde el command windows por el ususario y después de haber introducido los parámetros previamente definidos por él mismo.

4.1.2. Variables de la función llamada manualmente.

4.1.2.1. Variable ***q.***

También llamada variable articular y es obtenida por la función *genvar*, es un arreglo vectorial simbólico donde los componentes corresponden a los tipos de articulaciones que tiene cada uno de los *n* eslabones del robot. De igual manera, se puede decir que q_i es la posición de la *i*-ésima articulación.

4.1.2.5. Variable ***varq.***

Esta variable también es obtenida por *genvar y es* otro arreglo simbólico que corresponde a las primeras derivadas con respecto del tiempo de cada componente del arreglo *q*, por lo que se puede decir que es la velocidad de cada articulación del manipulador.

4.1.2.6. Variable ***vvarq.***

Vvarq es un vector de componentes simbólicas correspondientes a las segundas derivadas con respecto del tiempo del arreglo de q y se relaciona con la aceleración de las juntas del robot, por lo general, su valor numérico es nulo.

4.1.2.7. Variable ***M.***

Esta variable se genera luego de haber llamado la función *com* y representa la matriz M de orden nx4, donde n es el número de articulaciones del robot y las tres primeras columnas representan los centros de masa de los elementos, y la última columna la masa de los eslabones; cabe destacar, que si existe una carga en el efector final el centro de masa del último eslabón se obtiene mediante la ecuación 2.11 del capítulo II y la masa será la suma de la masa de él mismo mas la masa de la carga que sostiene.

4.1.3. Parámetros que debe introducir el usuario en el comand windows.

Luego de darle a conocer a MATLAB de qué tipo es cada articulación del robot y antes de hacer algun llamado a las funciones o scripts que se señalarán posteriormente, se procede a generar manualmente en el command windows los parámetros DH, éste sera una matriz donde cada fila corresponde a una

articulación, por lo que el número de filas es variable, y cada columna corresponde a los parámetros theta (primera columna), d (segunda columna), a (tercera columna) y alfa (cuarta columna), lo que sugiere que el tamaño de las columnas es fijo, entonces se puede definir que los parámetros DH es una matriz de orden nx4 y se trata simbólicamente debido a que la variable theta*i* tendrá el valor de q*i* si la articulación *i* es rotacional sino d*i* será quien tome el valor de q*i*.

Después, el usuario debe generar la matriz Mr de orden nx4, las filas dependen del número de articulaciones que posea el robot y las cuatro columnas representan: las tres primeras son las componentes X, Y y Z de los centros de masa de cada elemento referidos al sistema de coordenadas de cada articulación, y la útima, la masa de cada eslabón; también se debe introducir el vector fila Ml de orden 4 que corresponde, las tres primeras componentes al centro de masa de la carga sujeta al efector final del robot referida al sistema de coordenadas del última articualción y la cuarta componente es la masa de dicha carga. Después, se genera el vector *g* de orden 4, los tres primeros componentes son las coordenadas X, Y y Z de la fuerza de gravedad con el sistema de referencia en el origen del manipulador y la última será una componente nula. Para que se guarden todos estos datos en el workspace de MATLAB, el usuario debe darle una variable a cada dato que introduce, además posee la libertad de darle cualquier nombre, pero es recomendable a modo de entendimiento, de colocarle variables relacionadas con la teoría estudiada para el caso (ej.: para la matriz de parámetros de Denavit-Hartenberg guardarla bajo el nombre de DH y así con los demás datos).

4.1.4. Función que es utilizada sólo dentro de otra función.

4.1.4.1. FUNCIÓN ***mph***

Esta función consiste en proporcionar al usuario cada una de las matrices de paso homogénea (mph) de un robot; como en todas las funciones, se verifica el número de entradas posibles para la ejecución correcta de la misma, en caso contrario enviará al usuario un mensaje de error indicando cuál es el problema. El número de entradas permisibles en esta función es de uno (1), tres (3) y cuatro (4)

entradas; en caso de ser una es porque se esta introduciendo completamente la matriz Denavit-Hartenberg y la matriz resultante será la mph entre cero y el numero de articulaciones del robot, es decir desde el origen hasta el efector final del mismo que se le llama matriz de transformación homogénea; si el usuario introduce tres entradas tiene la posibilidad de elegir cuál es la mph que desea, es decir, se origina la matriz de paso homogénea de una parte o de la matriz completa de los parámetros D-H. Si son cuatro entradas, se esta colocando cada fila de la matriz D-H y sólo se genera la matriz mph de esa articulación referida a la articulación anterior. Además, con esta función, y sólo para el caso de tres (3) argumentos de entrada, se puede crear la matriz de paso homogénea inversa, es decir, en vez de que se genere la matriz ${}^{i}A_{j}$ se genera ${}^{j}A_{i,}$ como se utilizará en algoritmos posteriores.

La ecuación 2.7 es la que se toma en cuenta para la obtención de la mph y es por ello que dentro de esta función se hace llamados a tres funciones mas que son FUNCIÓN rotx, FUNCIÓN rotz y FUNCIÓN trasl.

Para la validación del número de entradas en ésta y las demás funciones a explicar, se utiliza una función que posee MATLAB llamada *nargin* , simplemente lo que hace es comparar el número de argumentos de entrada con la que se requiera en la función donde se esté aplicando.

Por último, en el llamado de esta *mph* dentro de otra función se utiliza como entrada tres argumentos, debido a que para efectos de uso en dicha función se necesita saber especificamente cuál es la matriz de transformación que se requiere.

4.1.4.2. FUNCIÓN ***rotx***

Rotx es una matriz homogénea de orden 4 producto de la rotación alrededor del eje x; solo acepta una entrada, alfa, que será el ángulo de rotación en grados, puede ser tratado simbólica o numéricamente, en caso de que se introduzca mas entradas se produce un mensaje de error. La ecuación con la que se determina rotx es la 2.4 del marco teórico.

4.1.4.3. FUNCIÓN *rotz*

Esta es una función que retorna una matriz homogénea de orden 4, producto de la rotación alrededor del eje z; el número de entrada permisible es uno, theta, en caso contrario la función arrojará un mensaje de error. Theta es el ángulo de rotación en grados y puede ser tratado numérica o simbólicamente. La ecuación 2.6 es la que se utiliza para la obtención de la rotación en el eje Z.

Para éstas dos últimas funciones se utilizaron *sin* y *cos* en lugar de *sinD* y *cosD* debido a que en éstas últimas no se permite el tratamiento simbólico de una variable.

4.1.4.4. FUNCIÓN *trasl*

El resultado de esta función es una matriz de orden 4, producto de la traslación en x, y, z. Se puede introducir una (1) o tres (3) entradas para obtener el resultado que se espera, de lo contrario, la función arrojará un mensaje de error. Si se introduce una entrada se tomará como un vector desplazamiento en el espacio, si se introduce tres entradas es porque se esta metiendo las coordenadas de traslación X, Y y Z. La ecuación 2.1 es la que se utiliza en esta función para determinar la traslación del robot.

4.1.5. Función utilizada dentro del script.

4.1.5.1. FUNCIÓN *uij*

La función uij retorna un arreglo matricial de matrices de variaciones de las matrices de paso $^{0}A_{i}$ respecto de cada una de las variables articulares q_j, el usuario puede introducir tres parámetros de entradas que son 'DH' que es una matriz que contiene los elementos de Denavit-Hartenberg donde la primera fila corresponde con la primera articulación y así sucesivamente, 'Q' es un vector de J elementos con las variables simbólicas articulares y N es el numero de articulaciones del robot, es opcional y por ello se puede admitir dos parámetros de entrada y N se calcula dentro de la función como la longitud de las columnas de la matriz DH. En caso de que las entradas de la función difieran de dos y tres parámetros, ésta arrojará un mensaje de error. Para el desarrollo de esta función

se tomó en cuanta la ecuación 2.15, pero la derivada de la matriz ºAi respecto a la coordenada qj puede obtenerse de manera computacional mediante la siguiente expresión:

$$\frac{d^0\mathrm{Ai}}{d\mathrm{qj}} = \begin{matrix} {}^0\mathrm{A}_{\mathrm{j}-1}\mathrm{Q}_\mathrm{j}{}^{\mathrm{j}-1}\mathrm{A}_\mathrm{i} & \text{si j} \le \text{i} \\ 0 & \text{si j} > \text{i} \end{matrix}$$

donde

$$Q_j = \begin{matrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$ si la articulación es rotacional.

$$Q_j = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{matrix}$$ si la articulación es prismática.

Por último, por efectos de programación del script del algoritmo, esta función se guarda en el workspace de matlab bajo el nombre de U.

4.1.5.2. FUNCIÓN *uijk*

Es una función similar a la anterior con la diferencia de que ésta retorna un arreglo vectorial de *n* componentes, cada fila corresponde a una matriz *U* que en cada columna se deriva por cada variable articular q_j. Requiere como argumentos de entrada la matriz U, producto de la función anterior, *q* y n, aunque ésta última no es obligatoria ya que dentro de la función se puede determinar; por lo que se puede concluir que para la ejecución correcta de esta función el número de entradas es de dos (2) o tres (3) argumentos. En el worskspace se guarda con el nombre de UK.

Para la segunda derivada de la matriz ºAi respecto a la coordenada qj puede obtenerse a través de la ecuación 2.16 y de manera computacional con la siguiente expresión:

$$\frac{d^0\mathrm{Ai}}{d\mathrm{qj}} = \begin{matrix} {}^0\mathrm{A}_{\mathrm{j}-1}\mathrm{Q}_\mathrm{j}{}^{\mathrm{j}-1}\mathrm{A}_{\mathrm{k}-1}\mathrm{Q}_\mathrm{k}{}^{\mathrm{k}-1}\mathrm{A}_\mathrm{i} & \text{si i} \ge \text{k} \ge \text{j} \\ {}^0\mathrm{A}_{\mathrm{k}-1}\mathrm{Q}_\mathrm{k}{}^{\mathrm{k}-1}\mathrm{A}_{\mathrm{j}-1}\mathrm{Q}_\mathrm{j}{}^{\mathrm{j}-1}\mathrm{A}_\mathrm{i} & \text{si i} \ge \text{j} \ge \text{k} \\ 0 & \text{si k} > \text{i o j} > \text{i} \end{matrix}$$

4.1.5.3. FUNCIÓN *ji*

La función *ji* es un arreglo vectorial de matrices de pseudoinercias, es decir que cada componente *i* del vector de salida corresponde a la matriz de pseudoinercia de la *i*-ésima articulación del robot. Para la ejecución correcta de esta función se necesitan dos (2) o tres (3) argumentos de entrada, los cuales son: la matriz M y el vector articular q, y opcional n, que es el número de articulaciones del manipulador; en caso contrario, se imprimirá un mensaje de error para el usuario.

Para la obtención de las matrices del vector J, se tomó en cuenta la ecuación 2.17 el marco teórico.

4.1.5.4. FUNCIÓN *d*

Ésta es una función que muestra la matriz simétrica inercial del robot. Para la obtención de esta matriz se necesita como argumentos de entrada la matriz de matrices U, el arreglo vetorial de matrices J y de manera opcional n, por lo que se define como necesario dos (2) o tres (3) variables de entrada en esta función. Se tomó en cuenta la ecuación 2.18 para obtener esta matriz de inercias, que se puede representar de la siguiente manera:

$$D = d\ i,j\ = \begin{matrix} d_{11} & \cdots & d_{1j} \\ \vdots & \ddots & \vdots \\ d_{i1} & \cdots & d_{ij} \end{matrix}$$

4.1.5.5. FUNCIÓN *hikm*

La salida es un arreglo vectorial de matrices cuyos términos están definidos por la ecuación 2.19, es necesario como argumentos de entrada la matriz de matrices U, también UK y y el vector de matrices J, lo que resume que para esta función se requiere de tres (3) o cuatro (4) variables de entrada, sino se imprime un mensaje de error para el usuario.

4.1.5.6. FUNCIÓN *hi*

Esta función, de acuerdo con la ecuación 2.20**,** posee como parámetros de entrada al vector de matrices calculado con la función HIKM, el vector simbólico de

las derivadas de *q* (*varq*) y el número de elementos del robot, siendo este último opcional. La salida consta de un vector relacionado con la velocidad donde cada elemento corresponde a las fuerzas de coriolis y centrípetas no lineal de los eslabones del robot, los componentes de H son:

$$H = h\ q, varq\ = (h_1, h_2, \dots, h_n)^T$$

4.1.5.7. FUNCIÓN *c*

Los argumentos de entrada son parámetros previamente definidos por el usuario (DH, M, g) y la matriz *U* generada anteriormente a través de la función *uij*, éstas son las entradas necesarias para la ejecución correcta de la función, es opcional introducir el valor de n debido a que dentro de la función se calcula en caso de su omisión como entrada, es decir sólo acepta cuatro (4) o cinco (5) entradas. A la salida se entrega un vector C donde cada elemento corresponde a la matriz columna de la fuerza de la carga gravitatoria, el cual se obtiene mediante la ecuación 2.21 dela marco teórico y se puede representar de la siguiente manera:

$$C = c\ q\ = (c_1, c_2, \dots, c_n)^T$$

4.1.6. Script del algoritmo

4.1.6.1. SCRIPT ***lageu***

Este script no es más que el desarrollo computacional del algoritmo de Langrage-Euler estudiado en capítulos anteriores; en él, sólo se hacen llamados a las funciones explicadas arriba y lo que genera es las tres (3) variables necesarias que son la matriz de fuerzas de inercias (D), la matriz de las fuerzas coriolis y centrípetas (H) y la matriz columna de fuerzas de gravedad (C) y la ecuación del par o fuerza aplicada en las articulaciones para producir el movimiento del manipulador particular planificado (τ) y se obtiene a través de la ecuación 2.22.

El desarrollo de las funciones anteriores, se realizó con la finalidad de utilizarlas dentro de este script, es decir que para conseguir esas tres (3) variables, en el command Windows de matlab no es necesario hacer los llamados de las

funciones, esto sólo se hace si el usuario desea hacer alguna verificación de cualquier función. En los scripts no se necesita algún argumento de entrada, basta con el llamado del mismo para su ejecución y que los datos previamente definidos por el usuario estén guardados en el workspace del MATLAB.

Ahora bien, si el usuario desea hacer una verificación del funcionamiento de alguna función, puede hacer el llamado de la misma en el command windows pero con el conocimiento previo de cuáles son los argumentos de entrada que se necesitan y comprobar que los mismos estén ya guardados en el workspace; en caso de que no estén allí, en el command Windows se generara un mensaje de error señalando que algún argumento de la función no está definido.

4.2. DESARROLLO DEL ALGORITMO COMPUTACIONAL DE NEWTON-EULER

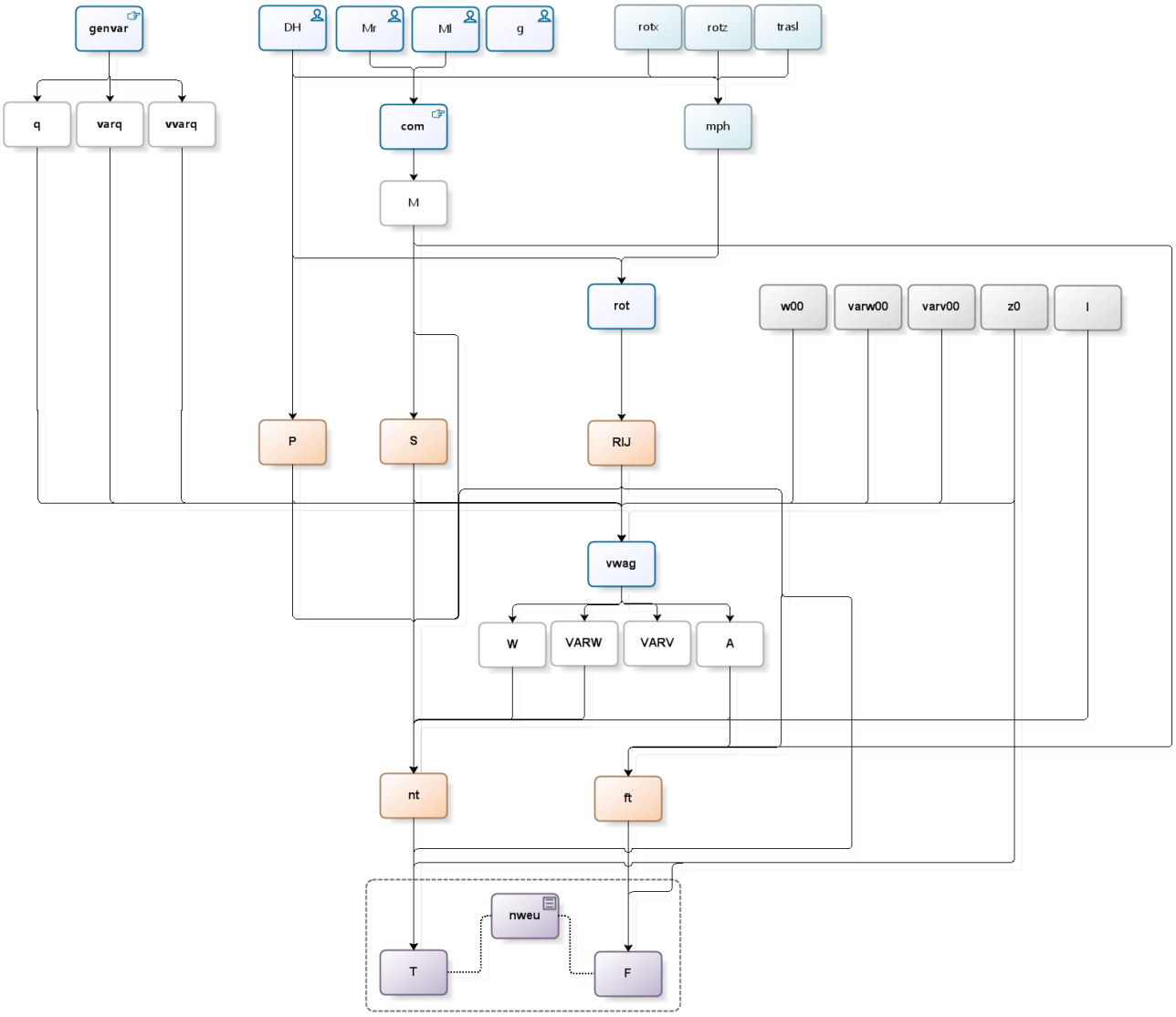


Fig. 4.2.- Algoritmo computacional de Newton-Euler.

Donde:

: Función que debe ser llamada manualmente en el comand windows por el usuario.

 : Parámetros que debe introducir el usuario en el comand windows.

 : Función utilizada en el script.

 : Variables que genera una función.

 : Función que es utilizada sólo dentro de otra función.

 : Inicializaciones de variables en el script.

 : Variables creadas en el script.

 : Script del algoritmo.

 : Variables de salida del script.

4.2.1. Función que debe ser llamada manualmente en el comand windows.

Como en el caso del algoritmo anterior, las funciones que son llamadas por el usuario son FUNCIÓN *genvar* y FUNCIÓN *com*, por lo que no es necesario repetir la funcionalidad de cada una en este punto.

4.2.2. Parámetros que debe introducir el usuario en el comand windows.

Estos parámetros son: la matriz DH, el centro de masa y la masa de los eslabones (matriz Mr), el centro de masa y la masa de la carga sostenida en el efector final del robot (matriz Ml) y el centro de gravedad (g), ya mencionados anteriormente en el desarrollo del algoritmo anterior.

4.2.3. Función utilizada en el script

4.2.3.1. FUNCIÓN ***rot***

Es una función que retorna una matriz de rotación no homogeneizada entre dos articulaciones; como argumentos de entrada, necesita los parámetros D-H, i, j, siendo estos dos últimos el número de la articulación j respecto a la anterior (i), entre los cuales se requiere la matriz de rotación. En ésta se hace uso de la función ***mph*** para la obtención de dicha matriz, a través de la sub matriz que describe la orientación del manipulador que se expresa en la ecuación 2.3 del marco teórico.

4.2.3.2. FUNCIÓN ***vwag***

Como parámetros de entrada de esta función se tiene z0, P, S, RIJ, q, varq, vvarq, que son previamente obtenidos en el script, así como otros argumentos de base fija en la programación del algoritmo, como lo son w00, varw00 y varv00 que es la aceleración lineal. La función *vwag* arroja como salida a las variables *W*, *VARW*, *VARV* y *A*.

4.2.4. Variables que genera una función

Algunas de las variables que se tratan aquí, son las generadas por la función *genvar* las cuales son: *q*, *varq* y *vvarq*. Otras variables son las que genera la función *vwag* y *rot*, éstas se pueden describir de la siguiente manera:

4.2.4.1. Variable ***W***

La variable *W* es una de las salidas de la función *vwag* y es un vector fila que representa la velocidad angular del sistema S_i, ésta depende de qué tipo es la articulación si es prismática o de revolución para que tome un valor como se observa en la ecuación 2.24 del marco teórico.

4.2.4.2. Variable ***VARW***

Ésta es otra salida de la función *vwag* que representa el vector fila de la aceleración angular del sistema S_i; para su obtención, también se tomó en cuenta el tipo de articulación que posee el manipulador y la ecuación 2.25 del capitulo II.

4.2.4.3. Variable ***VARV***

Es la aceleración lineal del sistema S_i, se obtuvo a través de la ecuación 2.26 del marco teórico y se representa de manera vectorial.

4.2.4.4. Variable ***A***

A es una variable de salida que representa el vector fila de la aceleración lineal del centro de gravedad del eslabón i, se obtuvo en base a lo expuesto en la ecuación 2.27 del marco teórico.

4.2.4.5. Variable ***RIJ***

Esta variable será una matriz de matrices de rotación y sus respectivas inversas, para la obtención de las matrices directas se hace un llamado de la función *rot* y para sus inversas basta con que las directas sean traspuestas, esto es debido a afirmaciones teóricas.

4.2.5. Función que es utilizada dentro de otra función

En este apartado las funciones a mencionarse son: *rotx*, *rotz*, *trasl* y *mph*, que son funciones que han sido explicadas anteriormente. En el desarrollo de este algoritmo, el funcionamiento y uso de ellas es exactamente igual que en el algoritmo anterior.

4.2.6. Inicializaciones de variables en el script

En el desarrollo del algoritmo, se ha necesitado la inicialización de ciertas variables como lo son: *w00*, *varw00* y *varv00*, las cuales son típicamente nulas salvo que la base del robot esté en movimiento. Además, se inicializa *Z0*, que es un vector unitario en dirección al eje de la articulación, y la variable *I* que es la matriz de inercia del eslabón i respecto de su centro de masa expresado en el sistema S_i, que para efectos de esta programación es una matriz nula, por tomarse en cuenta masas concentradas en cada elemento del manipulador. Estas inicializaciones se realizan dentro del desarrollo del script, es decir, que para algún cambio en ellas, el usuario necesita introducirse en el archivo .m del algoritmo que se ha creado y modificarlas.

4.2.7. Variables creadas en el script

En este desarrollo computacional de Newton-Euler, se necesita de la mención de estas variables para el progreso y entendimiento del mismo debido a que se necesitan para obtener otras variables de envergadura en este algoritmo.

4.2.7.1. Variable ***P***

Esta variable representa las coordenadas x_i, y_i y z_i de la localización del origen del sistema S_i respecto al sistema anterior (S_{i-1}) y se visualiza como un vector fila de tres (3) componentes; por último, se tomó en cuenta la ecuación 2.23 del marco teórico y de allí se dedujo que sólo se necesita de los parámetros D-H y operaciones matemáticas para su obtención computacional.

4.2.7.2. Variable ***S***

S es una matriz que representa las coordenadas del centro de masas del eslabón i respecto del sistema S_i; por lo que se acaba de mencionar, esta variable se puede obtener tomando las tres (3) primeras componentes de cada fila de la matriz M.

4.2.7.3. Variable ***ft***

La variable *ft* está definida como la fuerza ejercida sobre el elemento i del robot y para su obtención depende de la matriz de rotación, la masa (m_i) y aceleración A del mismo eslabón y de la fuerza ejercida sobre el eslabón siguiente a él, en el caso de cuando se está estudiando el elemento que esta unido al efector final se toma esta fuerza como nula. La ecuación para obtener *ft* es la 2.28 del capítulo II. Esta variable se obtiene si y sólo si la articulación i es prismática.

4.2.7.4. Variable ***nt***

La variable *nt* es el par o torque ejercido sobre el elemento i del manipulador y para su obtención se debió de definir o calcular con anterioridad los siguientes parámetros: RIJ, P, S, mi, A, I, VARW, W y el par ejercido sobre el eslabón siguiente; y como en la variable anterior, si el elemento en estudio es el unido al efector final se tomará como torque del siguiente como nulo. Esta variable

sólo se calcula si la articulación i es de revolución y se obtiene mediante la ecuación 2.29 del marco teórico.

4.2.8. Script del algoritmo

4.2.8.1. SCRIPT ***nweu***

El algoritmo computacional de Newton-Euler se desarrolla para un conjunto compacto de ecuaciones recursivas hacia adelante y hacia atrás que se les puede aplicar secuencialmente a los elementos del manipulador, es decir, que en este script se hace un llamado a la función *vwag* para el desarrollo de recursión hacia adelante, proporcionándole al usuario la información cinemática del manipulador desde la base hasta el efector final del mismo a través de sus salidas, las cuales son: *W*, *VARW*, *VARV* y *A*. Luego, realiza la recursión hacia atrás, ya no con un llamado de alguna función, sino proporcionándole una variable a cada ecuación estudiada con anterioridad, estas variables son las fuerzas y pares ejercidos sobre cada elemento desde el efector final hasta el sistema de referencia de la base y a partir de dichas variables se calculan los torques y fuerzas ejercidas sobre las articulaciones del robot.

Ahora bien, para la ejecución correcta de estas ecuaciones recursivas los parámetros definidos por el usuario deben estar guardados en el workspace de MATLAB, se inicializan las variables que se explicaron con anterioridad y se realiza el cálculo de *RIJ*, estos dos (2) últimos se hace dentro del algoritmo.

Las variables unidas a este elemento en el diagrama de bloques, *F* y *T*, son las salidas o variables que muestra este script como resultado final y se obtienen mediante la ecuación 2.30 del marco teórico, tomará la letra que le corresponda por el tipo de cada articulación del robot, si es F es por la fuerza ejercida por la articulación prismática, de lo contrario, si toma la variable T, es porque la salida de esa articulación es el torque ejercido sobre ella por ser rotacional.

Por otra parte, como se mencionó en el script ***lageu***, no es necesario algún argumento de entrada, sólo el llamado del mismo: ***nweu***, para su ejecución. Además, si el usuario requiere hacer uso de alguna de las dos (2) funciones

utilizadas, basta con llamarlas en el comand windows con el conocimiento previo de que se ha verificado que los argumentos de entradas se encuentren definidos antes de realizar dicho llamado, en caso contrario se imprime en pantalla un mensaje de error.

Por último, se puede visualizar al hacer el llamado de estos scripts que los dos arrojan los mismos resultados de la misma forma sólo que *lageu* tiene un tiempo de ejecución mayor en comparación con el tiempo de ejecución de *nweu* debido a la complejidad computacional de las ecuaciones del movimiento de los mismos; además, las salidas de la *lageu* son la matriz D, el vector columna C y H y el torque tl que se obtuvo de acuerdo a lo especificado en la ecuación 2.22, y las salidas de *nweu* son las fuerzas o pares de cada articulación y el torque total.

CAPÍTULO V

5.1. DESARROLLO DE LA INTERFAZ GRÁFICA DE USUARIO

MATLAB cuenta con una herramienta llamada GUIDE, para crear Interfaces Gráficas de Usuario (GUI); las GUI-s son ventanas muy útiles debido a que ayudan a la interacción hombre-máquina. La creación de estas interfaces en este Trabajo de Grado es de mucho beneficio para la adquisición de conocimientos del usuario, y la estructura que se diseñó para el entendimiento y exploración de estas interfaces es la mostrada en la figura 5.1.

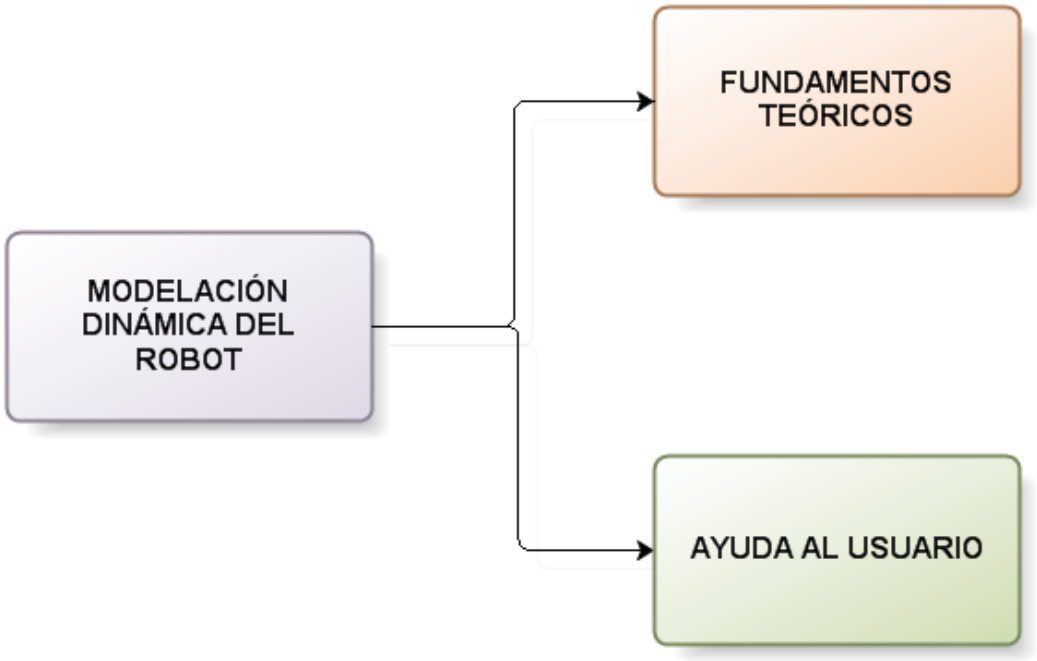


Figura 5.1.- Desarrollo de las Interfaces Gráficas de Usuario **(**GUI's).

Donde:

 : Conjunto de Interfaz Gráfica de Usuario

Con el propósito de facilitar la comprensión del manejo de la interfaz gráfica, se proyectaron tres (3) tipos de ventanas:

- Ventanas principales: El contenido de éstas llevan a que el usuario pueda hacer la modelación dinámica del manipulador y si desea, evaluar algunas de las funciones desarrolladas para los modelos. Son dieciocho (18) en total y se organizan como se explica más adelante.

- Ventanas de ayuda: En el recorrido de las ventanas principales, se notó que éstas necesitaban de otras que explicaran de una manera detallada cómo se usan, es por ello que se realizó la interfaz de ayuda la cual contiene el nombre de cada ventana principal y al activarla surge una explicación sobre cómo explorar y hacer un buen uso de la GUI principal. Para introducirse en éstas, se creó un menú ayuda en las ventanas principales, al hacer click en la palabra AYUDA, inmediatamente se abre la interfaz.
- Ventanas teóricas: El contenido de las ventanas principales puede resultar desconocido para algunos de los usuarios que desee explorarlas, por esta razón se realizaron interfaces que solvente este problema. Las GUI's tienen los fundamentos teóricos necesarios para el conocimiento básico sobre robots industriales y modelación dinámica de los mismos; ellas se pueden abrir para su exploración de dos formas: la primera, haciendo click en el menú fundamentos teóricos creado en las interfaces principales y la segunda, a través del icono de ayuda que poseen algunos botones o paneles en las mismas; luego de tener alguna interfaz teórica abierta se puede explorar con los botones atrás y siguiente que tienen cada una de ellas.

5.1.1. DESARROLLO DE LA INTERFAZ (GUI's) PRINCIPAL

En el diseño de las ventanas principales, se tomó en cuenta la estructura mostrada en la fig. 5.2, allí se observa que una interfaz precede a otra o a varias de acuerdo a lo que se requiera. En la exploración de las mismas se puede llegar a varios objetivos, como lo son: la modelación dinámica de un manipulador seleccionado a través de dos formulaciones, objetivo principal de este trabajo, y a la evaluación numérica de algunas funciones que se crearon para el desarrollo computacional de los algoritmos.

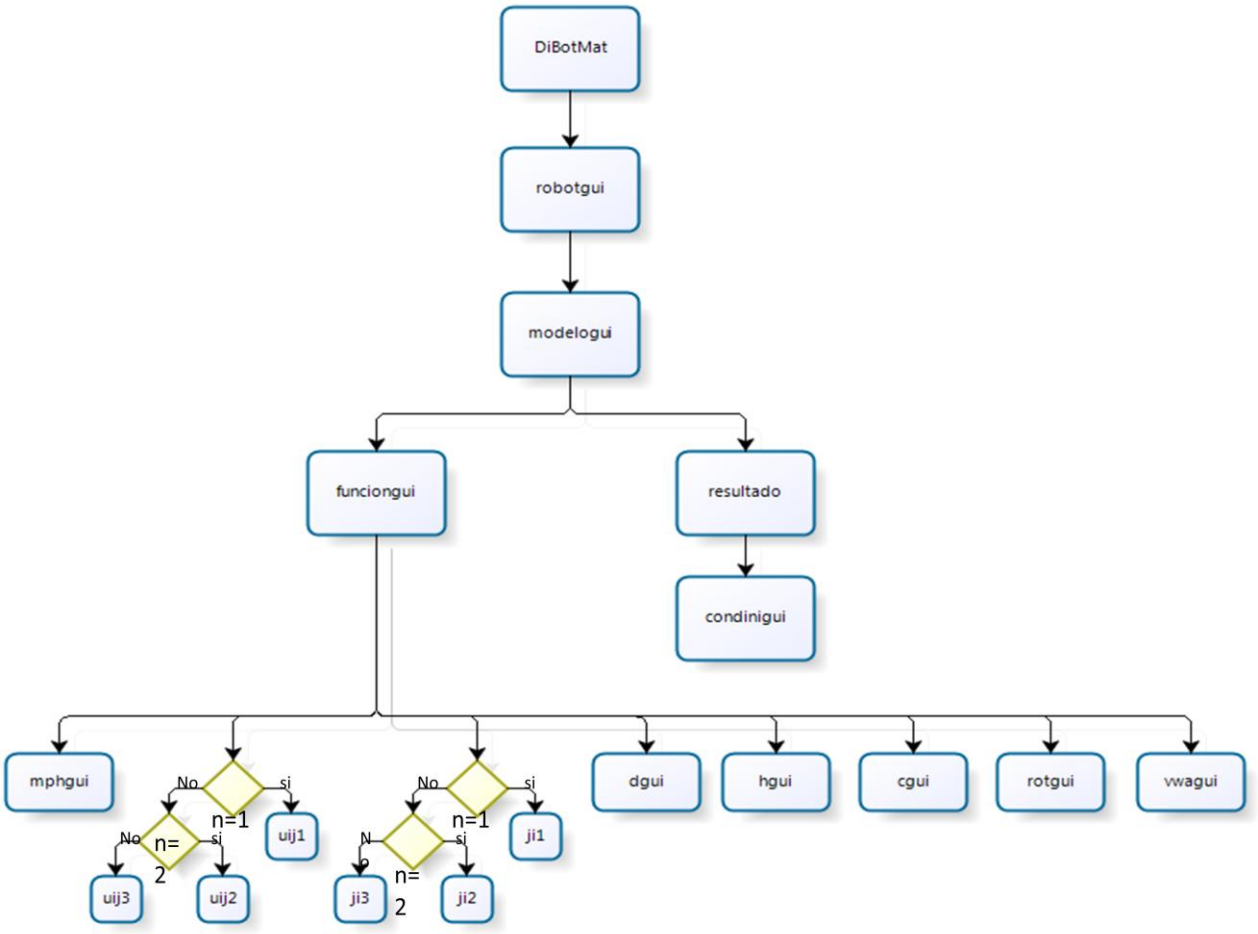


Fig. 5.2. Desarrollo de las ventanas principales.

5.1.1.1. *DiBotMat*

Es la primera ventana de las interfaces principales, por lo que se puede decir que es la presentación de este Trabajo de Grado y en ella se visualiza el título del mismo que lleva por nombre *"DiBotMat"*, modelación dinámica de robots industriales, además de los autores y da paso a la próxima interfaz, como se observa en la fig. 5.3. Cabe recordar, que ésta y las GUI's que continúan, tienen una ayuda de cómo explorar cada una de ellas, tal como se muestra en la fig. 5.4; así como los fundamentos teóricos necesarios para que el usuario conozca de manera resumida los conceptos básicos del contenido que se trata en estas interfaces, ver fig. 5.5. Además, esta barra de menú posee la herramienta de abrir

el manual de usuario que ha sido anexado en este trabajo (fig. 5.6), el cual contiene un ejemplo que ayuda a conocer estas interfaces.

Fig. 5.3. Interfaz Gráfica de Usuario *DiBotMat*.

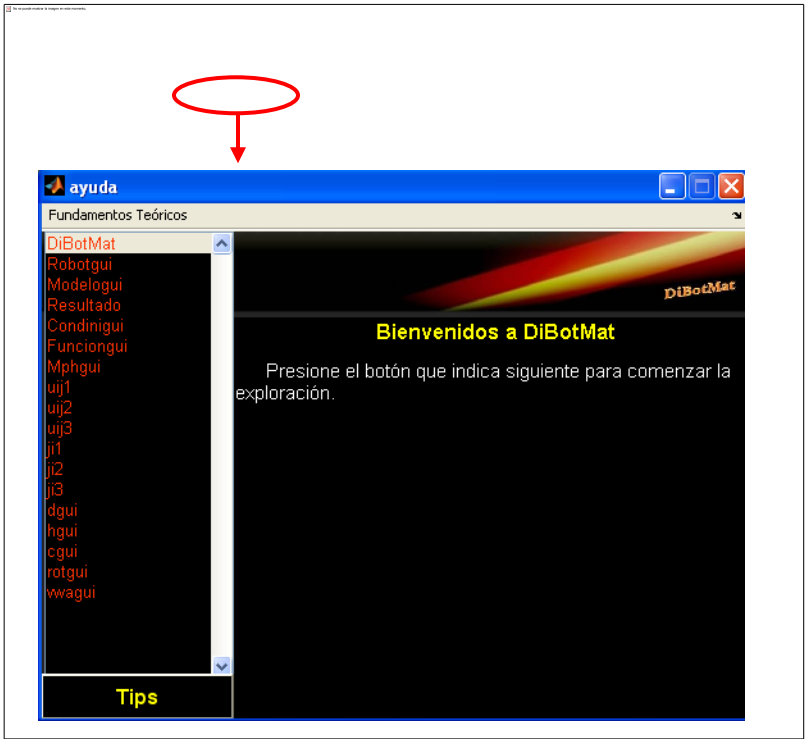


Fig. 5.4. Ayuda para el uso de las ventanas principales.

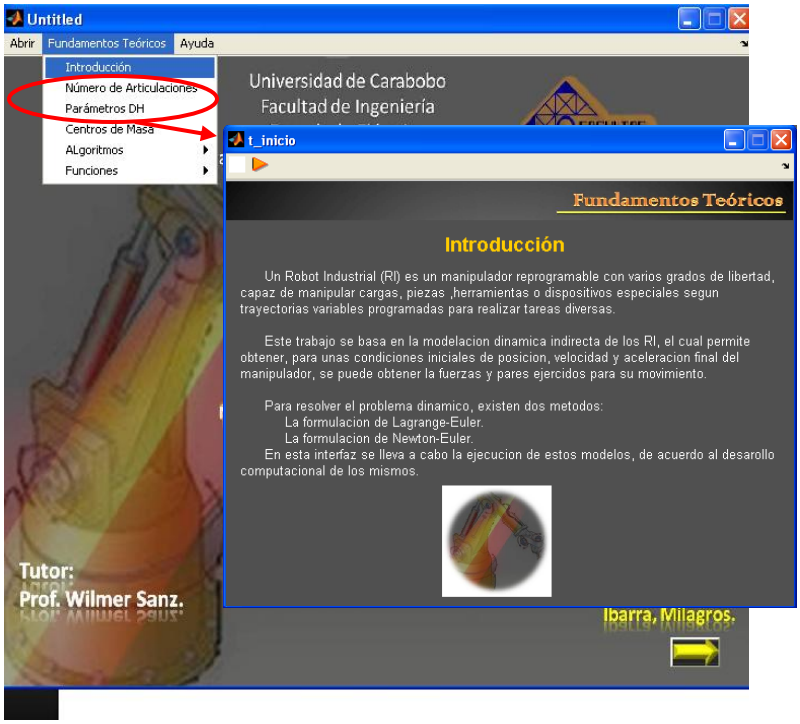


Fig. 5.5. Fundamentos Teóricos sobre modelación dinámica.

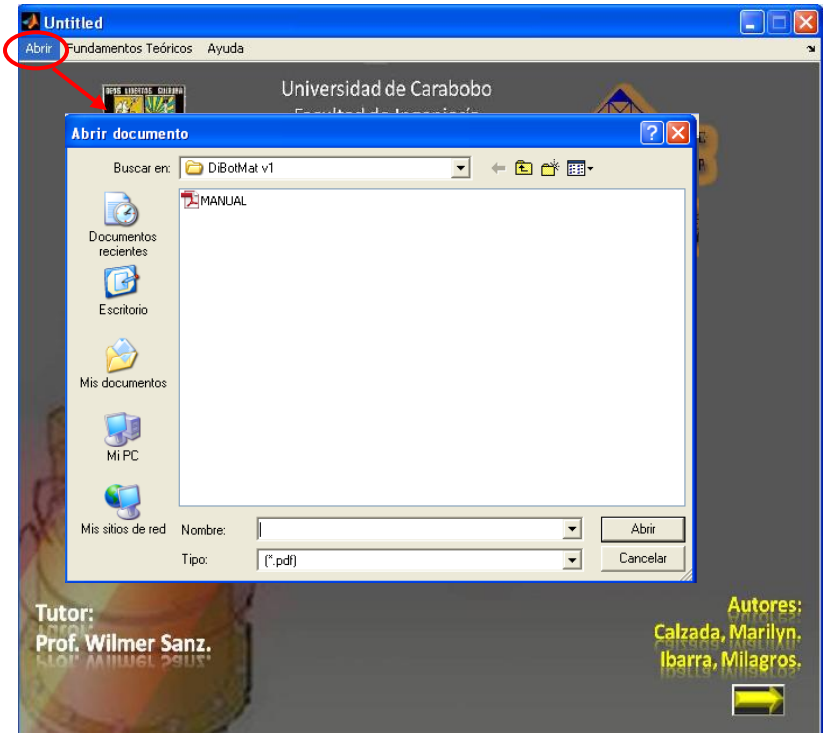


Fig. 5.6. Herramienta para abrir el manual de usuario.

5.1.1.2. *Robotgui*

Para el inicio de la modelación en MATLAB se necesitan los parámetros que debe definir el usuario; por ende, se diseñó la ventana llamada *robotgui* con una distribución de su espacio como se muestra en la fig. 5.7; se puede decir que es una de las GUI-s más importantes, en la cual se deben introducir los datos correctamente del robot deseado; ésta es una de las ventanas que posee el icono de ayuda en todos los paneles que tiene.

Para el desplazamiento entre ventana y ventana hay botones indicativos de atrás y adelante que lleva al usuario a la anterior y siguiente respectivamente.

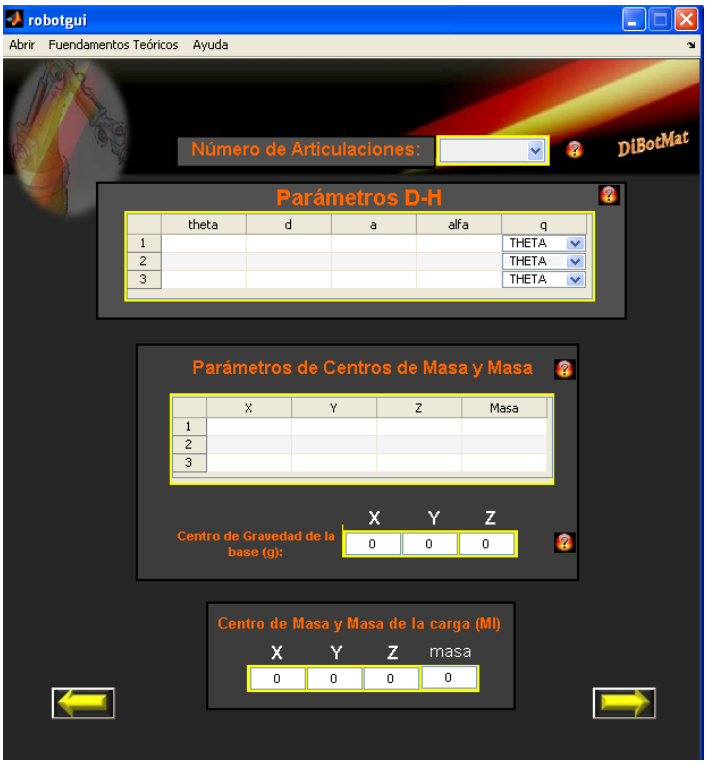


Fig. 5.7. Interfaz Gráfica de Usuario *robotgui*.

5.1.1.3. *Modelogui*

En el desarrollo de esta ventana se agruparon en paneles, como se observa en la fig 5.8, las tareas que puede ejecutar el usuario. Son dos paneles, uno contiene dos botones, cada uno con los algoritmos para la modelación dinámica Lagrange-Euler y Newton-Euler, se puede oprimir uno o los dos modelos; el otro

panel posee un botón con el nombre de *función*, éste al presionarlo inmediatamente lleva a otra ventana. En cambio, para el panel descrito al principio de este punto, se debe presionar el botón siguiente para trasladarse a otra interfaz, sino se ha realizado acción alguna y se presiona siguiente, se verá en pantalla un mensaje de error que le indicará que no ha presionado ningún botón de este panel.

Al presionar cualquier algoritmo, se genera en el *command windows* y se guarda en el *workspace* las ecuaciones simbólicas de los mismos y en la interfaz, dentro del panel, se mostrará el tiempo de ejecución, indicándole así al usuario que ya puede presionar el botón siguiente.

Fig. 5.8. Interfaz Gráfica de Usuario *modelogui*.

5.1.1.4. *Resultado*

Esta interfaz fue creada como se muestra en la fig. 5.9, con la finalidad de visualizar los resultados, de acuerdo al algoritmo anteriormente seleccionado por el usuario en la ventana de *modelogui*, luego de presionar el botón *introducir condiciones iniciales*, es decir, si sólo se presionó un modelo, se observará un

vector fila correspondiente al resultado de dicha formulación; en cambio, si en *modelogui* se oprimió los dos algoritmos, en la ventana *resultado* se verá el resultado de ambos.

Esta GUI arroja el objetivo principal de este desarrollo computacional, por lo que sólo se puede retroceder a la ventana anterior.

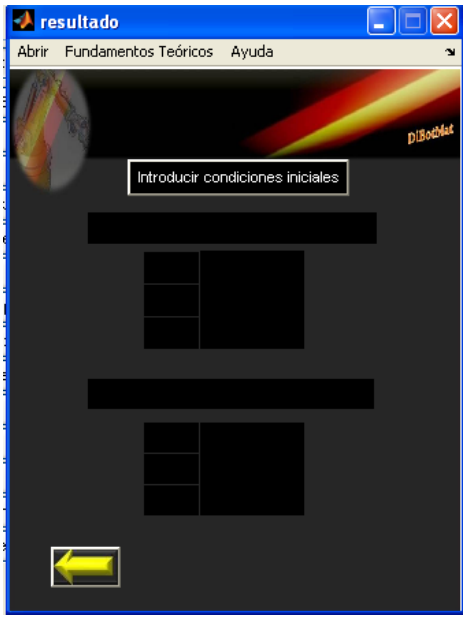


Fig. 5.9. Interfaz Gráfica de Usuario *resultado*.

5.1.1.5. *Condinigui*

Esta interfaz se creó con el propósito de introducir las condiciones iniciales para evaluar las ecuaciones simbólicas de los dos algoritmos, como se observa en la fig. 5.10. Esta ventana se abre cuando se oprime el botón *introducir condiciones iniciales* expresado en el ítem anterior y el tamaño de filas de la tabla será de acuerdo al número de articulaciones, al presionar OK retorna a la gui *resultado*.

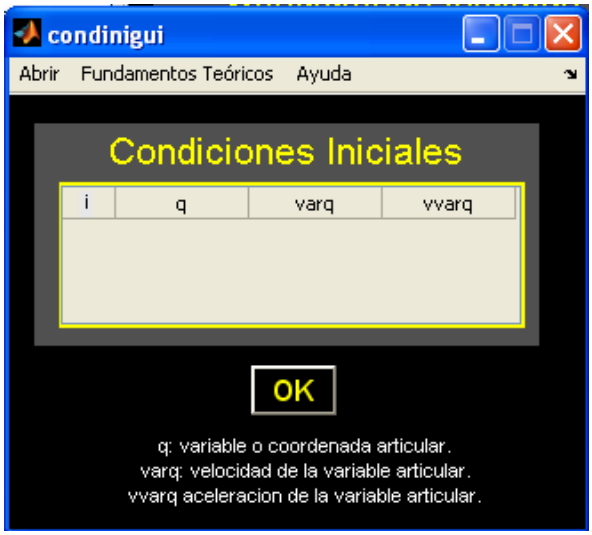


Fig. 5.10. Interfaz Gráfica de Usuario *condinigui*.

5.1.1.6. *Funcióngui*

Con el propósito de tener la opción de evaluar las funciones creadas para los algoritmos, se diseñó la ventana *función*; esta interfaz ofrece las funciones más resaltantes o claves para el desarrollo computacional de las formulaciones, ellas se muestran en botones agrupados en paneles de acuerdo a su uso en cada modelo. En cada botón se colocó un icono de ayuda en caso de que el usuario sienta alguna inquietud sobre lo que describe él mismo, esto se realizó con la finalidad de ir de manera un poco más directa a la respuesta de la duda teórica que se tenga, de igual forma, posee el menú de los fundamentos teóricos y ayuda correspondiente al uso de esta ventana.

Función posee sólo botón para retroceder, cada botón de los paneles llevará al usuario a la ventana correspondiente para evaluar esa función. El diseño de esta interfaz se observa en la fig. 5.11.

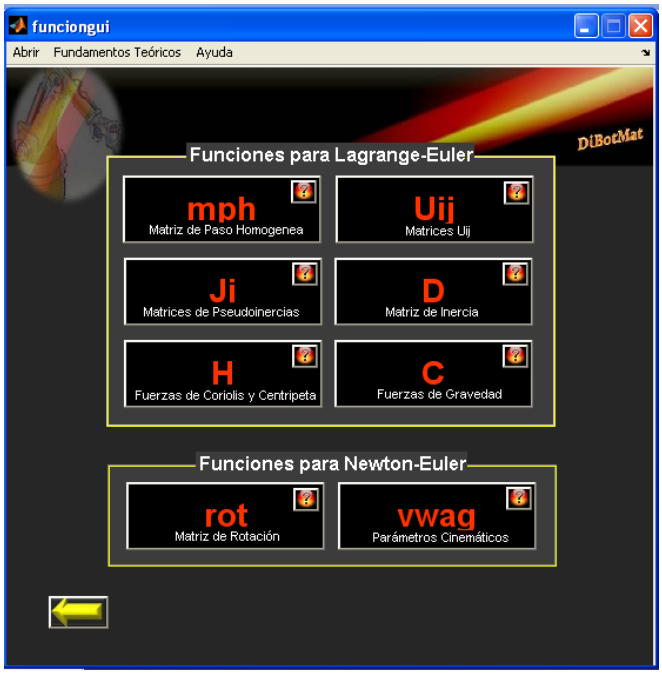


Fig. 5.11. Interfaz Gráfica de Usuario *función*.

5.1.1.7. *Mphgui*

La interfaz *mphgui* no es más que una ventana donde se observa la matriz de transformación homogénea (mph), está diseñada de forma muy práctica permitiendo al usuario fácil interactividad y dominio de la GUI con el propósito de visualizar la mph que se desee simplemente introduciendo los parámetros necesarios, que se definen cuáles son al abrirse esta ventana, luego de tener éstos se presiona el icono que representa igual. La interfaz resultante es la mostrada en la figura 5.12.

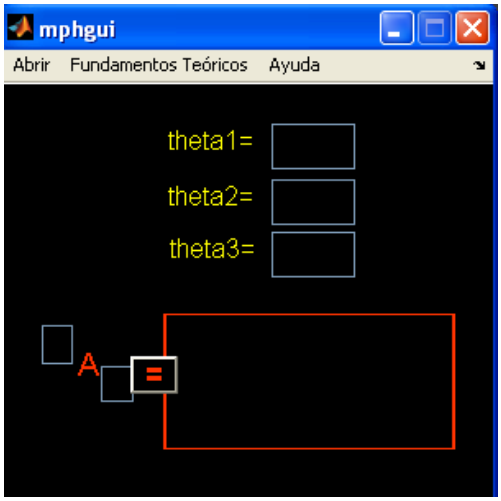


Fig. 5.12. Interfaz Gráfica de Usuario *mphgui* para n=3.

5.1.1.8. *Uij1*

Para el desarrollo de cómo mostrar las matrices “uij” explicada en capítulos anteriores, se toma en cuenta el número de articulaciones del manipulador debido a que si incrementa éste más serán las matrices “uij” a visualizar; por esta razón y para el aprovechamiento del espacio, esta ventana muestra la matriz para un robot con una sola junta. Esta interfaz es muy simple sin mucha complejidad y con la estructura de la mayoría de las ventanas donde se evalúan las funciones, la cual es una parte para introducir condiciones iniciales y otra para observar el resultado, posee un botón el cual genera la matriz resultante. El diseño de esta GUI se muestra en la fig. 5.13.

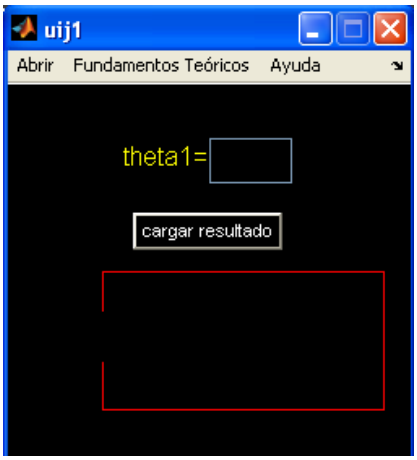


Fig. 5.13. Interfaz Gráfica de Usuario *uij1*.

5.1.1.9. *Uij2*

Uij2 arroja las matrices “uij” para robots de dos (2) articulaciones, las GUI-s que nos muestran estas matrices están diseñadas para que cuando el usuario introduzca el número de articulaciones en la interfaz *robotgui* y en caso de querer evaluar la función uij, abra automáticamente la ventana correspondiente con el número de articulaciones ante seleccionados, indicándole cuáles son las condiciones iniciales que necesita para evaluar. Para el caso de ésta y las ventanas que evalúan las funciones mencionadas en *funcióngui*, no poseen botones de continuación o retorno, sólo se cierra y queda a la vista del usuario la GUI antes mencionada. El diseño de esta interfaz se muestra en la fig. 5.14.

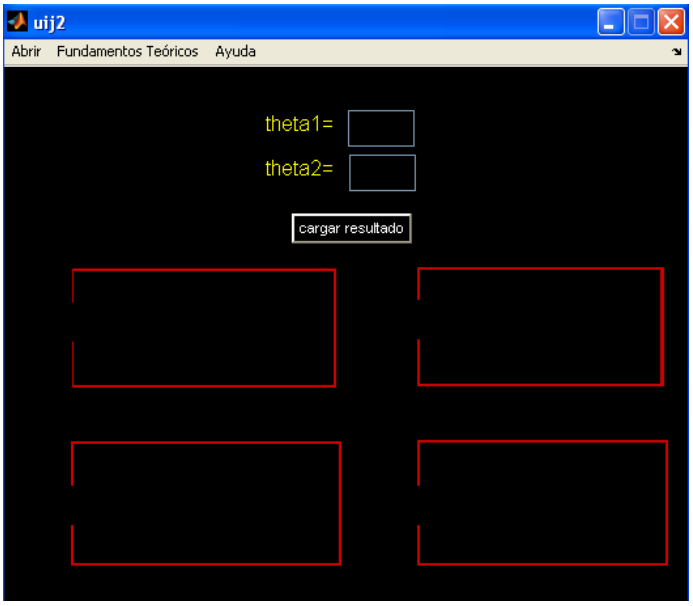


Fig. 5.14. Interfaz Gráfica de Usuario *uij2*.

5.1.1.10. *Uij3*

La ventana uij3 será vista por el usuario sólo cuando el robot estudiado o diseñado por él sea de tres (3) articulaciones; es una ventana diseñada como se observa en la fig. 5.16 y donde se visualizan las matrices “uij” presentes en el robot, es de gran tamaño debido a la cantidad de matrices presentes para el caso, las cuales son nueve (9) matrices “uij”. Como en la teoría estudiada en capítulos anteriores se expresa que éstas son matrices de una matriz, se quiso diseñar la visualización de ellas de esa manera, es decir, como se muestra en la figura 5.15.

Esta interfaz igual presenta la estructura de otras funciones, parte donde se introduce datos y parte donde se visualiza los resultados.

$$\begin{bmatrix} [U_{11}] & [U_{12}] & [U_{13}] \\ [U_{21}] & [U_{22}] & [U_{23}] \\ [U_{31}] & [U_{32}] & [U_{33}] \end{bmatrix}$$

Fig. 5.15. Matrices “uij” para un robot de tres articulaciones.
Fuente: Propia

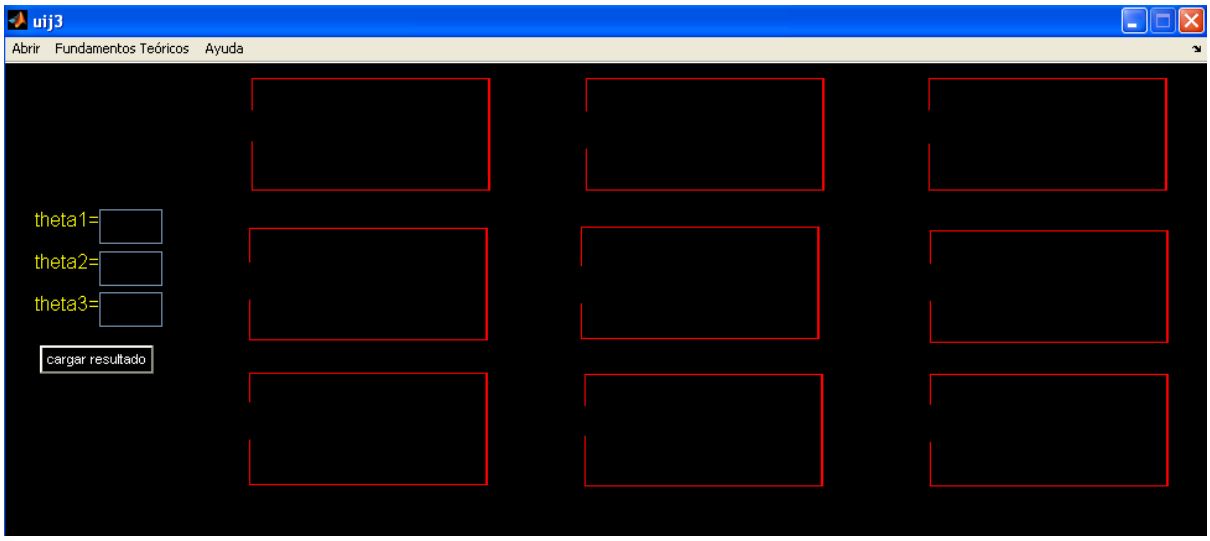


Fig. 5.16. Interfaz Gráfica de Usuario *uij3*.

5.1.1.11. *Ji1*

La interfaz *ji1* le suministra al usuario la matriz de pseudoinercia cuando el manipulador posee una sola articulación; su diseño se basó en mostrar la matriz, como se observa en la fig. 5.17, por ello se colocó un botón que debe presionar el usuario para poder visualizar el resultado, no necesita introducir condiciones iniciales para observar la matriz numérica.

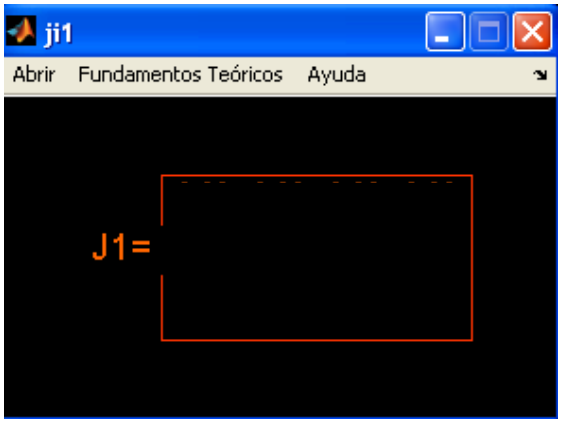


Fig. 5.17. Interfaz Gráfica de Usuario *ji1*.

5.1.1.12. *Ji2*

Esta ventana imprime en ella las dos (2) matrices de pseudoinercia correspondiente cada una a cada articulación del robot, por lo que se deduce que esta GUI se abre cuando el manipulador posee dos (2) grados de libertad. Esta interfaz tampoco necesita que el usuario introduzca condiciones iniciales, sólo presiona el botón mostrar resultado y aparece en pantalla las matrices ji. Su diseño se muestra en la fig. 5.18.

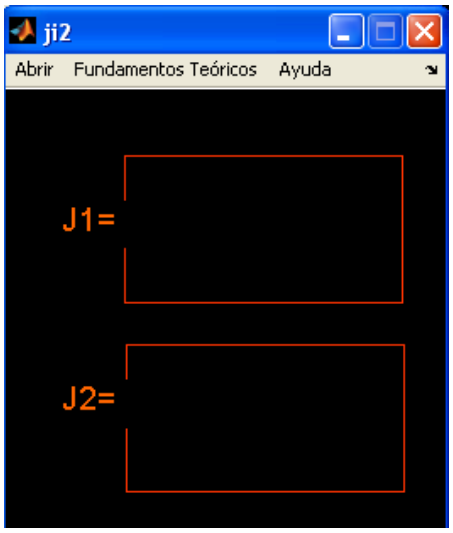


Fig. 5.18. Interfaz Gráfica de Usuario *ji2*.

5.1.1.13. *Ji3*

Ji3 muestra en su espacio, como se observa en la fig. 5.19, las matrices de pseudoinercia de un robot de tres (3) articulaciones; presenta la misma estructura

de visualización que las dos ventanas anteriores. Es necesario recordar, que si el usuario hace uso de la función ji, la interfaz de *funciongui* está programada para abrir la ventana correspondiente que muestra los resultados de ji, dependiendo del número de articulaciones.

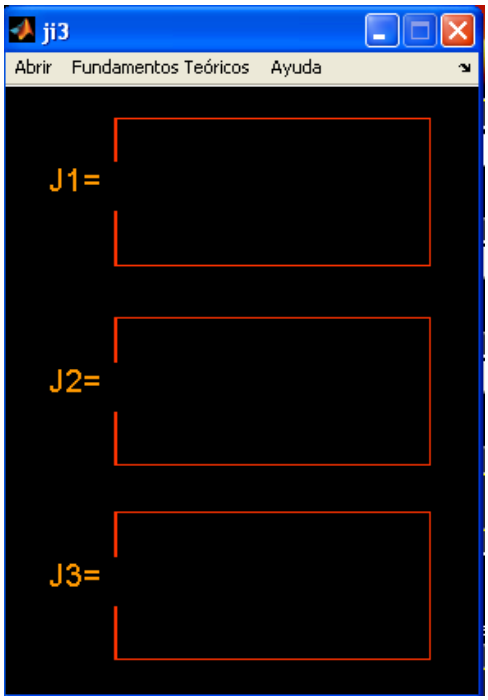


Fig. 5.19. Interfaz Gráfica de Usuario *ji3*.

5.1.1.14. *Dgui*

La ventana dgui, brinda total practicidad; su diseño, como se muestra en la fig. 5.20, permite conocer los resultados arrojados por la función d. Para su visualización en la GUI, el usuario no necesita realizar los cálculos o llamados de las funciones a través de la cuales se obtiene la matriz de inercia D, sólo introducir las condiciones iniciales y presionar mostrar resultado para ver la matriz resultante.

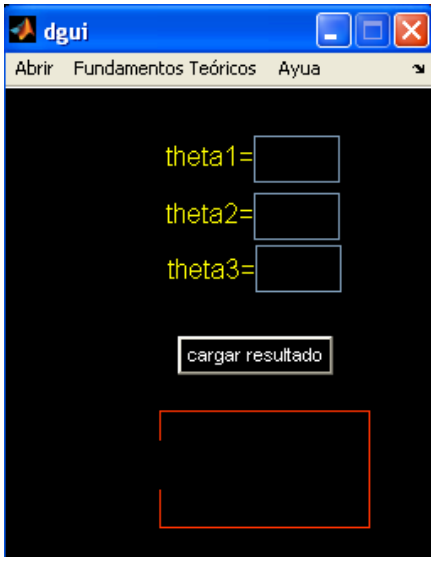


Fig. 5.20. Interfaz Gráfica de Usuario *dgui*.

5.1.1.15. *Hgui*

La interfaz *hgui* se abre cuando el usuario presiona en *funciongui* la función H, que representa el vector columna de las fuerzas de coriolis y centrípeta; en esta ventana se deben introducir las condiciones que se les señala para poder evaluar y mostrar la matriz resultante. Su diseño se observa en la fig. 5.21.

Fig. 5.21. Interfaz Gráfica de Usuario *hgui*.

5.1.1.16. *Cgui*

El diseño de cgui, como se muestra en la fig. 5.22 es de gran similitud al de la interfaz anterior; se basó en el mismo criterio, tratar de mostrar el resultado después de la evaluación de las ecuaciones simbólicas con ciertas condiciones iniciales introducidas por el usuario. El resultado a visualizar es la matriz columna de fuerzas de gravedad, no requiere ningún cálculo por medio de los usuarios de las funciones a través de las cuales se obtiene C.

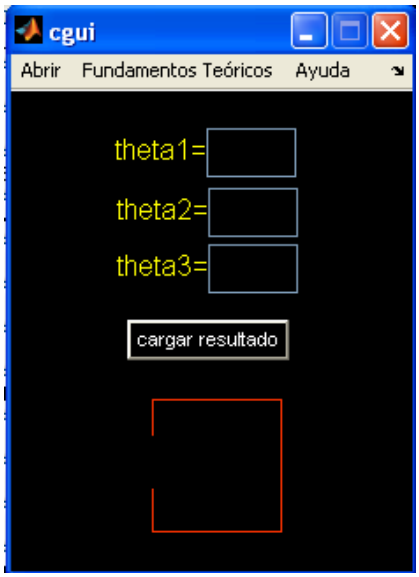


Fig. 5.22. Interfaz Gráfica de Usuario c*gui*.

5.1.1.17. *Rotgui*

Su diseño, como se muestra en la fig. 5.23, tiene parecido con la interfaz *mphgui*, esta ventana permitirá que el usuario obtenga la matriz de rotación de un robot, bastará definir las condiciones iniciales y entre cuáles articulaciones desea ver la rotación del manipulador, para así visualizar dicha matriz al presionar un botón con el símbolo de igualdad. Puede también, si lo desea, calcular la matriz de rotación inversa.

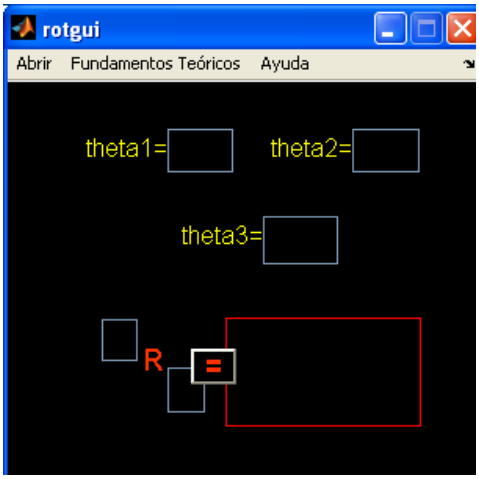


Fig. 5.23. Interfaz Gráfica de Usuario rot*gui*.

5.1.1.18. *Vwagui*

Esta GUI cuenta con una tabla de condiciones iníciales, la cual debe llenarse con los datos requeridos para dar paso a la obtención de los resultados; dicha GUI se creó con la finalidad de mostrar los parámetros cinemáticos del robot, los cuales son: velocidad y aceleración angular del sistema i, aceleración lineal del sistema i y aceleración lineal del centro de gravedad del eslabón i. El diseño de esta interfaz se observa en la fig. 5.24.

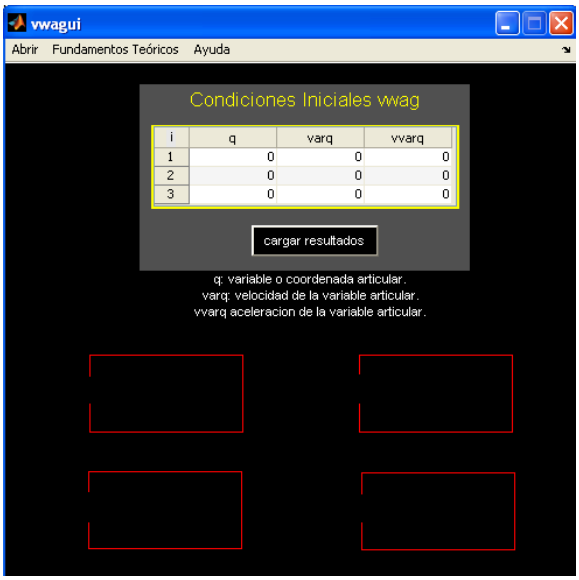


Fig. 5.24. Interfaz Gráfica de Usuario vwa*gui*.

CAPÍTULO VI

Este capítulo reflejará la validación del Trabajo de Grado mediante comparación de sus resultados con los que arroja otra herramienta semejante que se ejecuta en MATLAB v.6.5.0.180913a Release 13 bajo el sistema operativo Windows XP. La herramienta elegida para hacer la contrastación fue la *Planar Manipulator Toolbox*, PLANMANT de ahora en adelante para hacer referencia a ella, elaborada por León Zlajpah. Antes de estudiar el comportamiento de un robot en esta *Toolbox*, se debe realizar una pequeña explicación de las ecuaciones del movimiento empleadas que definen la dinámica de un manipulador planar de n grados de libertad con articulaciones rotacionales. Para ello primero se realiza un análisis cinemático del mismo; como se especifica a continuación.

6.1. ANÁLISIS CINEMÁTICO DE UN MANIPULADOR PLANAR n-R EN PLANMANT

Para el análisis cinemático de un manipulador, PLANMANT toma en cuenta las coordenadas articulares si están definidas como ángulos relativos o absolutos. Para el caso de este trabajo se tomó en cuenta las coordenadas de las juntas definidas como ángulos absolutos (fig. 6.1). [24]

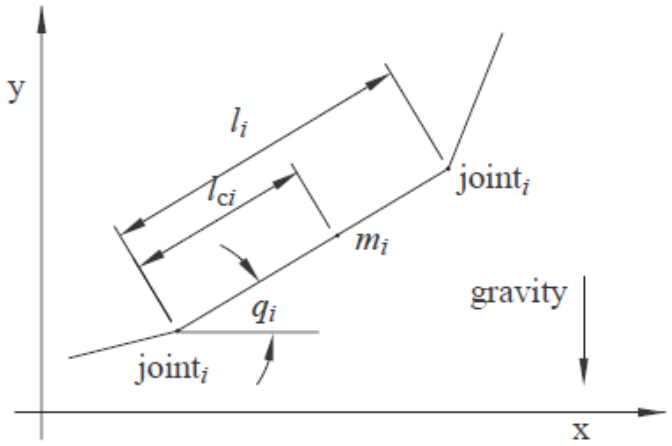


Fig. 6.1. Estructura de un manipulador planar de n grados de libertad con posiciones de las juntas definidas como ángulos absolutos. [24]

La posición del efector final es $\boldsymbol{x} = [x, y]^T$ y puede ser expresado de acuerdo a las siguientes ecuaciones:

$$x = \sum_{k=1}^{n} l_k \cos(q_k) \qquad y = \sum_{k=1}^{n} l_k \operatorname{sen}(q_k) \tag{6.1}$$

donde n es el número de articulaciones que posee el manipulador y l_k es la longitud del i-ésimo eslabón. Es importante aclarar que se supone que el manipulador está moviéndose en el plano *xy*. [24]

Luego se deriva la ecuación 6.1 (ecuación 6.2) con respecto a la variable articular para obtener las componentes de la matriz Jacobiana (J) (ecuación 6.3).

$$\frac{dx}{dq_k} = -l_k \operatorname{sen}\ q_k \qquad \frac{dy}{dq_k} = l_k \cos(q_k) \tag{6.2}$$

$$J = \begin{matrix} -l_1 \operatorname{sen}\ q_1 & \dots & -l_n \operatorname{sen}\ q_n \\ l_1 \cos(q_1) & \dots & l_n \cos(q_n) \end{matrix} \tag{6.3}$$

Ahora se deriva la matriz J con respecto al tiempo (ecuación 6.4).

$$J = \sum_{k=1}^{n} \frac{dJ}{dq_k} q_k = \begin{matrix} -l_1 \cos\ q_1\ \ q_1 & \dots & -l_n \cos\ q_n\ \ q_n \\ -l_1 \operatorname{sen}(q_1) q_1 & \dots & -l_n \operatorname{sen}(q_n) q_n \end{matrix} \tag{6.4}$$

Con las matrices anteriores se obtiene una matriz JJ^T (ecuación 6.5), la cual es una matriz simétrica de orden 2x2.

$$JJ^T = \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix} \tag{6.5}$$

Con las componentes definidas de la siguiente forma:

$$a_{11} = \sum_{k=1}^{n} l_k^2 \operatorname{sen}(q_k)^2$$

$$a_{12} = a_{21} = \sum_{k=1}^{n} -l_k^2 \operatorname{sen}(q_k) \cos(q_k)$$

$$a_{22} = \sum_{k=1}^{n} l_k^2 \cos(q_k)^2$$

Y sus respectivas derivadas con respecto a la variable articular quedan definidas de la siguiente manera:

$$\frac{da_{11}}{dq_k} = 2l_k^2 \sin\ q_k\ \cos(q_k)$$

$$\frac{da_{12}}{dq_k} = \frac{da_{21}}{dq_k} = l_k^2\ (\sin\ q_k\ ^2 - \cos\ q_k\ ^2)$$

$$\frac{da_{22}}{dq_k} = -2l_k^2 \sin\ q_k\ \cos(q_k)$$

6.2. ANÁLISIS DINÁMICO DE UN MANIPULADOR n-R EN PLANMANT.

El modelo dinámico está basado en la formulación Lagrangiana. Las ecuaciones del movimiento están dadas por un balance de energía (ecuación 6.6).

$$\frac{d}{dt}\frac{\partial T}{\partial q} - \frac{\partial T}{\partial q} = \tau \tag{6.6}$$

Donde T es la energía total del sistema y τ las fuerzas generalizadas correspondientes a las coordenadas articulares q. Después de algunos cálculos, las ecuaciones dinámicas de un manipulador pueden ser escritas como se expresa en la ecuación 6.7. [24]

$$\tau = H\ q\ .q + C\ q,q\ + B(q) + g(q) \tag{6.7}$$

Donde H es la matriz de inercia, h es el vector de la fuerzas de Coriolis, centrífuga y fricción viscosa y g vector de fuerza de gravedad. Con lo dicho anteriormente, se puede deducir que:

$$h\ q,q\ = C\ q,q\ + B(q)$$

La matriz de inercia H de la ecuación 6.7 viene dada por la expresión de la ecuación 6.8.

$$H = \ \sum_{i=1}^{n}(m_i {J_L^i}^T J_L^i + {J_A^i}^T I_i J_A^i \tag{6.8}$$

Donde m_i es la masa y I_i es la matriz de inercia y $J_L^{\ i}$ y $J_A^{\ i}$ son las matrices jacobianas para el centro de masa del i-ésimo enlace asociado con la velocidad lineal y angular respectivamente.

Para conocer las matrices jacobianas expresadas anteriormente se debe saber cuáles son las componentes del centro de masa, las cuales están definidas por la ecuación 6.9. [24]

$$x_{ci} = \begin{matrix} \left(\sum_{k=1}^{n} l_k \cos(q_k)\right) + l_{ci}\cos(q_i) \\ \left(\sum_{k=1}^{n} l_k \operatorname{sen}(q_k)\right) + l_{ci}\operatorname{sen}(q_i) \end{matrix} \tag{6.9}$$

Donde l_{ci} es la longitud entre la i-ésima articulación y el centro de masa del i-ésimo elemento. Derivando la ecuación 6.9 con respecto a q se obtiene la ecuación 6.10.

$$\frac{\partial x_{ci}}{\partial q_j} = \begin{matrix} -l_j sen\ q_j & si\ j < i \\ -l_{cj} sen\ q_j & si\ j = i \\ 0 & si\ j > i \end{matrix}$$

$$\frac{\partial y_{ci}}{\partial q_j} = \begin{matrix} l_j cos\ q_j & si\ j < i \\ l_{cj} cos\ q_j & si\ j = i \\ 0 & si\ j > i \end{matrix} \tag{6.10}$$

Ahora se deriva la ecuación 6.10 con respecto a la variable articular q (ecuación 6.11).

$$\frac{\partial^2 x_{ci}}{\partial q_j q_k} = \begin{matrix} -l_j cos\ q_j & si\ j = k\ \ j < i \\ -l_{cj} cos\ q_j & si\ j = k\ \ j = i \\ 0 & si\ j > i \end{matrix}$$

$$\frac{\partial^2 y_{ci}}{\partial q_j q_k} = \begin{matrix} -l_j sen\ q_j & si\ j = k\ \ j < i \\ -l_{cj} sen\ q_j & si\ j = k\ \ j = i \\ 0 & si\ j > i \end{matrix} \tag{6.11}$$

Empleando la ecuación 6.10 las matrices jacobianas asociadas con el centro de masa del i-ésimo eslabón son definidas como se muestra en la ecuación 6.12. [25]

$$J_L^{\ i} = \begin{matrix} -l_1 sen\ q_1 & \cdots & -l_{ci} sen\ q_1 & 0 & \cdots & 0 \\ l_1 sen\ q_1 & \cdots & l_{ci} sen\ q_1 & 0 & \cdots & 0 \end{matrix} \qquad (6.12)$$

En el caso de un manipulador planar con articulaciones rotacionales el término

$$J_A^{\ i\ T} I_i J_A^{\ i} = I_i \begin{matrix} I_{ixi} & 0 \\ 0 & 0 \end{matrix}$$

Ahora bien, se puede simplificar un poco mas el término anterior diciendo que la matriz H debido a la matriz de inercia de los elementos I_i viene dada por la siguiente expresión:

$$H_i = diag(I_i)$$

Usando la ecuación 6.10, las derivadas de J_L con respecto a la variable articular q son obtenidas se muestra en la ecuación (6.13).

$$\frac{\partial J_L^{(i)}}{\partial q_k} = \begin{matrix} 0 & \cdots & 0 & -l_k cos\ q_k & 0 & \cdots & 0 & si\ k < i \\ 0 & \cdots & 0 & -l_k sen\ q_k & 0 & \cdots & 0 & \\ 0 & \cdots & 0 & -l_{ci} cos\ q_k & 0 & \cdots & 0 & si\ k = i \\ 0 & \cdots & 0 & -l_{ci} sen\ q_k & 0 & \cdots & 0 & \\ 0 & & & & & & & si\ k > i \end{matrix} \qquad (6.13)$$

Con esta ecuación se procede a determinar el vector de fuerzas de coriolis, centrífuga y fricción viscosa (ecuación 6.14). [25]

$$h_i = \sum_{j=1}^{n} \sum_{j=1}^{n} h_{ijk} q_j q_k \qquad donde\ h_{ijk} = \frac{\partial H_{ij}}{\partial q_k} - \frac{1}{2} \frac{\partial H_{jk}}{\partial q_i} \qquad (6.14)$$

Y las componentes del vector fuerza de gravedad g para un robot planar de n grados de libertad vienen dadas por la ecuación 6.9.

$$g_i = \left(\sum_{k=i+1}^{n} m_k \right) l_i + m_i l_{ci} \cos(q_i) \qquad (6.9)$$

6.3. VALIDACIÓN DEL COMPORTAMIENTO DINÁMICO DEL MANIPULADOR PLANAR HASTA DOS GRADOS DE LIBERTAD.

Para comprobar la validez de este Trabajo de Grado se tomaron en cuenta diferentes casos, en los cuales se definen los parámetros de un manipulador de uno y dos eslabones y se calcula en DiBotMat el torque ejercido por los enlaces de cada robot seleccionado y luego, se procede a tomar este par e introducirlo en una implementación en Matlab/Simulink empleando la PLANMANT (fig.6.2). Así se verifica que el par calculado a través de este trabajo es el necesario para que el robot cumpla con la condición de equilibrio.

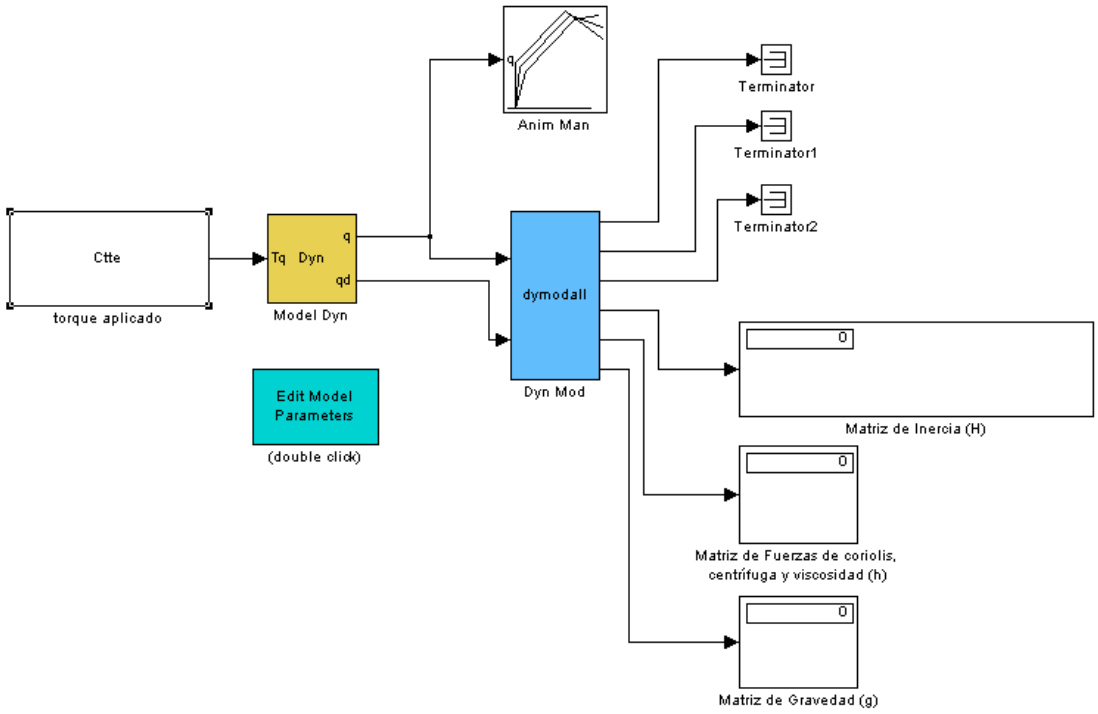


Fig. 6.2. Implementación en Matlab/Simulink empleando PLANMANT para el estudio dinámico de un manipulador planar.

En la implementación mostrada en la fig. 6.2 se muestra los bloques usados de la PLANMANT para la validación, uno de ellos es el Model Dyn el cual posee como parámetro de entrada el torque calculado en DiBotMat y como salida las

variables articulares y sus respectivas derivadas. El otro bloque importante a resaltar es el de edición de parámetros del manipulador, como lo son la longitud de los eslabones (L), longitud de los centros masa (Lc), masa de cada enlace (m), masa de la carga (ml), Inercia de los enlaces (II), coeficiente de fricción viscosa (Bv), posición inicial de los eslabones (q0) y velocidad inicial de los mismos (qd0), éstos pueden observarse en la fig. 6.3 que muestra cómo están expuestos dichos parámetros dentro de esta simulación. En la fig. 6.3 se observa que además de estos datos a introducir hay otros dos los cuales se calculan internamente en el bloque y ellos representan la posición inicial del efector final (x0) y su respectiva velocidad (xd0).

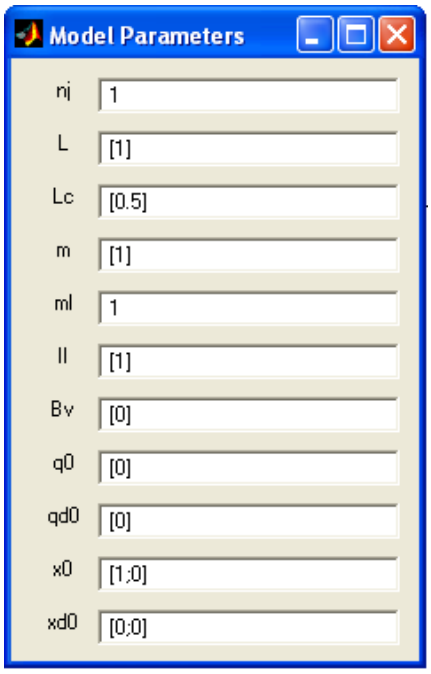


Fig. 6.3. Bloque para editar los parámetros del manipulador.

Otro bloque resaltante en esta implementación es el Dyn Mod que es un subsistema el cual posee como parámetros de entrada las salidas del bloque Model Dyn y genera la matriz de inercia (H), el vector de fuerzas de coriolis, centrífuga y viscosidad (h) y el vector de gravedad (g), los cuales para efectos de comparación en DiBotMat, serían D, H y C respectivamente calculados bajo el algoritmo de Lagrange-Euler.

Para comprobar el correcto funcionamiento de los bloques que conforman el sistema visto en la fig. 6.2, se analizaron diferentes casos de manipuladores ante varios torques de entrada que fueron previamente calculados en DiBotMat.

Caso 1.- Como primer caso de análisis se tomará un robot de un enlace y los datos del mismo son los mostrados a continuación:

En DiBotMat	En PLANMANT
$DH = \ q(1) \quad 0 \quad 1 \quad 0 \quad theta$	$L = [1]$
$Mr = \ -0.5 \quad 0 \quad 0 \quad 1$	$Lc = [0.5]$
$Ml = \ 0 \quad 0 \quad 0 \quad 0$	$m = [1]$
$g = \ 0 \quad -9.81 \quad 0 \quad 0$	$ml = [0]$
theta1=0	$II = [0.25]$
varTheta1=0	$Bv = [0]$
vvarTheta1=0	$q0 = [0]$
	$qd0 = [0]$

Donde algunos de los parámetros de DiBotMat se pueden expresar en función de los de PLANMANT de la siguiente manera:

$$DH = \ q(1) \quad 0 \quad L \quad 0 \quad theta$$

$$Mr = \ -Lc \quad 0 \quad 0 \quad m$$

$$Ml = \ 0 \quad 0 \quad 0 \quad ml$$

Al introducir los parámetros en DibotMat se obtiene el torque ejercido por el enlace de acuerdo al valor dado en la posición final de la variable articular (theta1=0) y su primera y segunda derivada (varTheta1=0 y vvarTheta1=0) las

cuales representan la velocidad y aceleración de la coordenada articular respectivamente. Este par calculado es el siguiente:

$$Por\ Lagrange - Euler: \quad tl = 4.905 = 9.81 * 0.5$$
$$Por\ Newton - Euler: \quad tn = 4.905 = 9.81 * 0.5$$

Cabe destacar que el cálculo del torque se puede realizar por alguno de los dos algoritmos, como se demuestra anteriormente, su valor es el mismo para ambas formulaciones y el que arroja la DiBotMat es el primer valor mostrado.

Ahora se introduce el segundo valor en la simulación usada en PLANMANT, como se observa en la fig. 6.4.

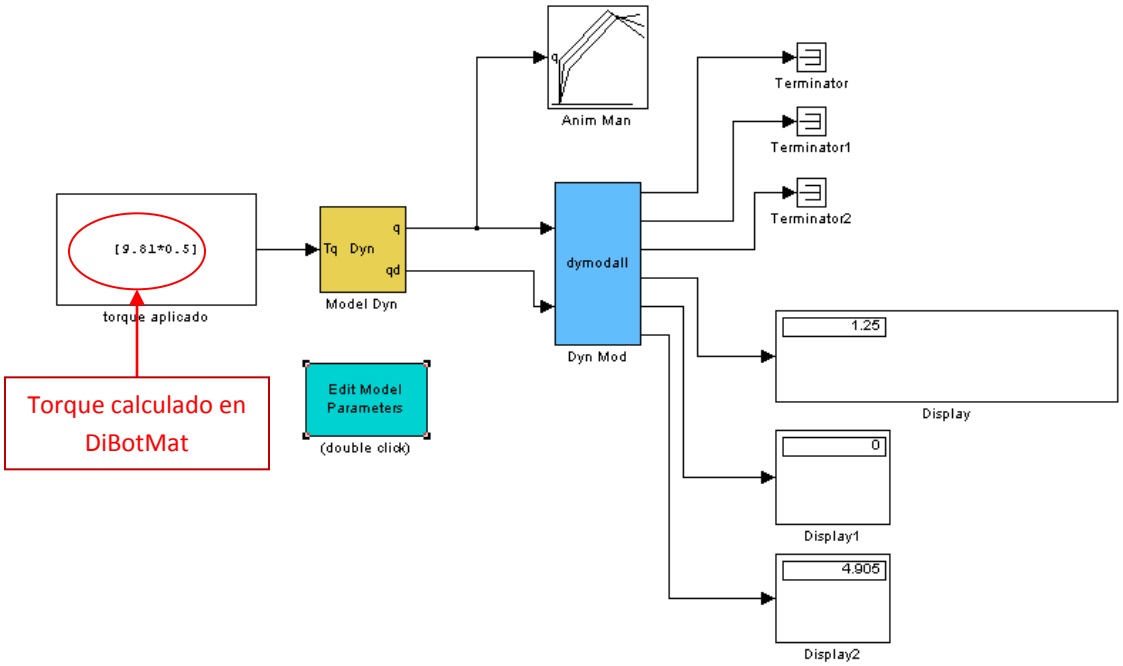


Fig. 6.4. Implementación en Matlab/Simulink para el estudio dinámico de un manipulador planar de un enlace ante un torque de entrada de 4.905 *Nw-m*.

Es importante resaltar que las matrices que se visualizan en la fig. 6.4 (H, h y g) son calculadas de una forma muy diferente en PLANMANT que en DiBotMat por lo que la matriz de inercia H es diferente a la matriz de inercia D, las otras si

son iguales en comparación con las de este trabajo a través de la formulación de Lagrange-Euler.

En la fig. 6.5 se puede apreciar la posición de equilibrio alcanzada por el brazo, es decir no realiza ningún movimiento debido a que el torque aplicado compensa la fuerza de gravedad que actúa en el brazo.

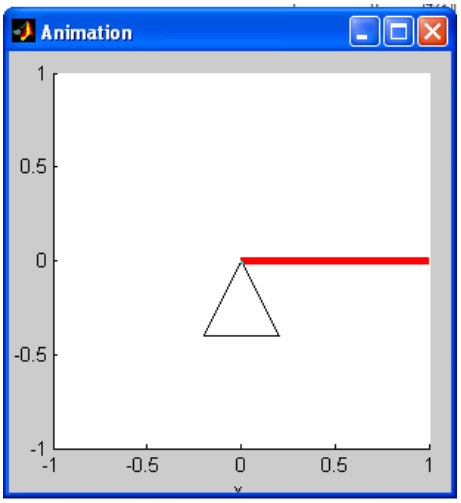


Fig. 6.5. Posición final del Manipulador (0°) de un enlace sin carga sostenida.

Caso 2.- Una variación del ejemplo previo se hará en este caso, donde la mayoría de los parámetros se mantienen igual sólo cambiará la posición final en DiBotMat y la posición inicial en PLANMANT.

En DiBotMat	En PLANMANT
$DH = \ q(1) \quad 0 \quad 1 \quad 0 \quad theta$	$L = [1]$
$Mr = \ -0.5 \quad 0 \quad 0 \quad 1$	$Lc = [0.5]$
$Ml = \ 0 \quad 0 \quad 0 \quad 0$	$m = [1]$
$g = \ 0 \quad -9.81 \quad 0 \quad 0$	$ml = [0]$
theta1=45	$II = [0.25]$

varTheta1=0	$Bv = [0]$
vvarTheta1=0	$q0 = [pi/4]$
	$qd0 = [0]$

Es necesario mencionar que los ángulos que se introducen en DiBotMat son en grados debido a que en su programación interna se realiza el cambio a radianes, pero en PLANMANT si se introduce en radianes.

El torque calculado en DiBotMat es el mostrado a continuación:

$$Por\ Lagrange - Euler: \quad tl = 3.4684 = 9.81 * 0.5 * \cos(^{pi}{}_{4})$$

$$Por\ Newton - Euler: \quad tn = 3.4684 = 9.81 * 0.5 * \cos(^{pi}{}_{4})$$

Ahora se introduce el segundo valor en la simulación usada en PLANMANT, como se observa en la fig. 6.6.

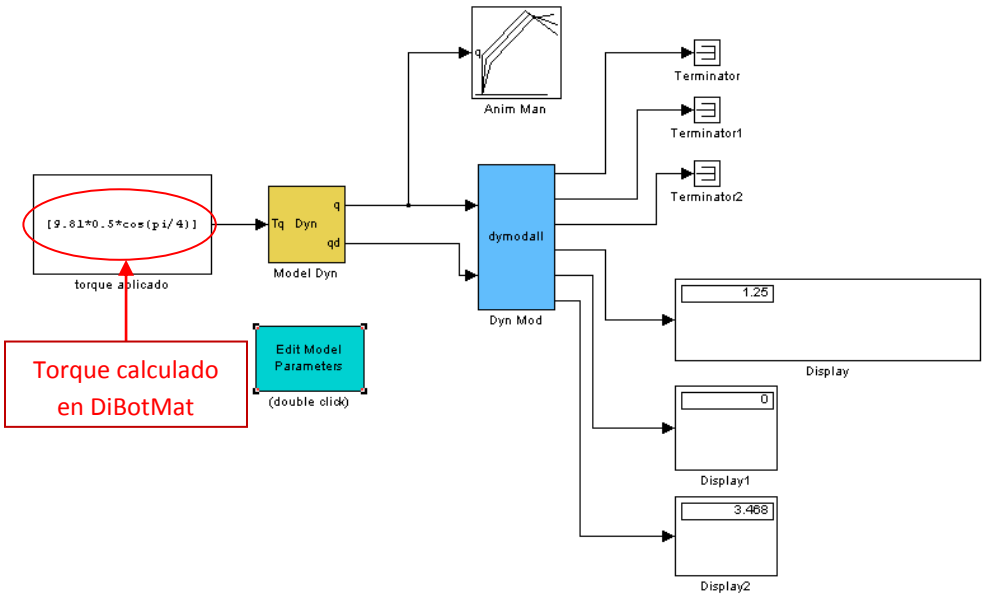


Fig. 6.6. Implementación en Matlab/Simulink para el estudio dinámico de un manipulador planar de un enlace ante un torque de entrada de 3.4684 *Nw-m*.

En la fig. 6.7 se puede apreciar la posición de equilibrio alcanzada por el brazo en la posición deseada (45°), es decir no realiza ningún movimiento debido a que el torque aplicado compensa la fuerza de gravedad que actúa en el brazo.

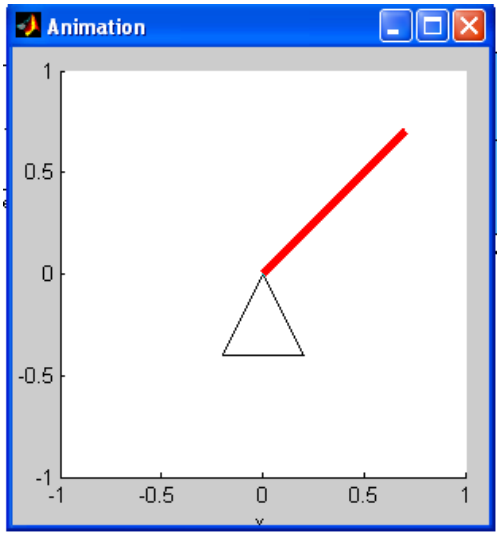


Fig. 6.7. Posición final del Manipulador (45°) de un enlace sin carga sostenida.

Caso 3.- En este tercer ejemplo se toma en cuenta el caso anterior pero con una carga sostenida en el efector final del manipulador. Los parámetros para definir el robot son:

En DiBotMat	En PLANMANT
$DH = \ q(1) \quad 0 \quad 1 \quad 0 \quad theta$	$L = [1]$
$Mr = \ -0.5 \quad 0 \quad 0 \quad 1$	$Lc = [0.5]$
$Ml = \ 0 \quad 0 \quad 0 \quad 1$	$m = [1]$
$g = \ 0 \quad -9.81 \quad 0 \quad 0$	$ml = [1]$
theta1=45	$II = [0.25]$
varTheta1=0	$Bv = [0]$

vvarTheta1=0 $q0 = [pi/4]$

$qd0 = [0]$

El torque calculado en DiBotMat es el que se muestra a continuación:

$$Por\ Lagrange - Euler: \quad tl = 10.4051 = 9.81 * 1.5 * \cos(^{pi}{}_{4})$$

$$Por\ Newton - Euler: \quad tn = 10.4051 = 9.81 * 1.5 * \cos(^{pi}{}_{4})$$

Ahora se introduce el segundo valor en la simulación usada en PLANMANT, como se observa en la fig. 6.8.

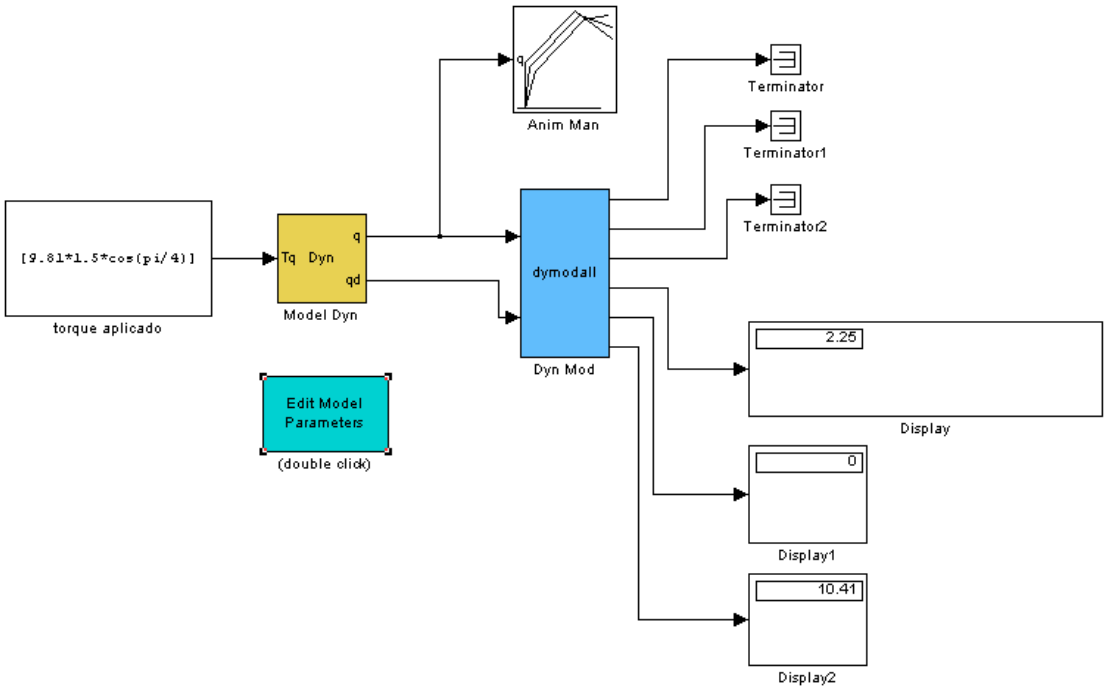


Fig. 6.8. Implementación en Matlab/Simulink para el estudio dinámico de un manipulador planar de un enlace con carga sostenida en el efector final ante un torque de entrada de 10.4051 *Nw-m*.

En la fig. 6.9 se puede observar que se alcanza la condición de equilibrio en el manipulador de un enlace cuando sostiene una carga en su efector final,

además se observa que el torque aumenta considerablemente para poder mantener la misma posición en la que se encontraba en el caso anterior.

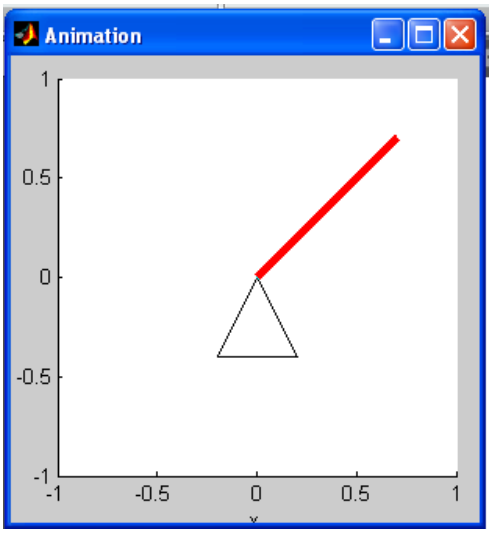


Fig. 6.9. Posición final del Manipulador (45°) de un enlace con carga sostenida.

Caso 4.- En este ejemplo se tomará como análisis de estudio un robot de dos enlaces y sus parámetros son:

En DiBotMat	En PLANMANT
$DH = \begin{matrix} q(1) & 0 & 1 & 0 & theta1 \\ q(2) & 0 & 1 & 0 & theta2 \end{matrix}$	$L = [1 \quad 1]$
$Mr = \begin{matrix} -0.5 & 0 & 0 & 1 \\ -0.5 & 0 & 0 & 0 \end{matrix}$	$Lc = [0.5 \quad 0.5]$
$Ml = \begin{matrix} 0 & 0 & 0 & 0 \end{matrix}$	$m = [1 \quad 0]$
$g = \begin{matrix} 0 & -9.81 & 0 & 0 \end{matrix}$	$ml = [0]$
theta1=0	$II = [0.25 \quad 0.25]$
varTheta1=0	$Bv = [0 \quad 0]$
vvarTheta1=0	$q0 = [0; 0]$

$$qd0 = [0; 0]$$

De manera que se entienda la relación entre algunos parámetros de las dos Toolboxes se muestra la siguiente leyenda:

$$DH = \begin{matrix} q(1) & 0 & l_1 & 0 & theta1 \\ q(2) & 0 & l_2 & 0 & theta2 \end{matrix} \qquad L = [l_1 \quad l_2]$$

$$Mr = \begin{matrix} -l_{c1} & 0 & 0 & m_1 \\ -l_{c2} & 0 & 0 & m_2 \end{matrix} \qquad Lc = [l_{c1} \quad l_{c2}]$$

$$Ml = \begin{matrix} 0 & 0 & 0 & ml \end{matrix} \qquad m = [m_1 \quad m_2]$$

$$ml = [m_l]$$

El torque calculado en DiBotMat es:

$$Por\ Lagrange - Euler: \quad tl = \begin{matrix} 4.905 \\ 0 \end{matrix} = \begin{matrix} 9.81 * 0.5 \\ 0 \end{matrix}$$

$$Por\ Newton - Euler: \quad tn = \begin{matrix} 4.905 \\ 0 \end{matrix} = \begin{matrix} 9.81 * 0.5 \\ 0 \end{matrix}$$

Como se ha realizado en los casos anteriores se introduce el segundo valor mostrado en la expresión anterior en la simulación creada en PLANMANT (fig. 6.2) y se demostrará que para este par aplicado se mantiene en la posición inicial (0; 0) tal como se observa en la fig. 6.10, es decir se cumple la condición de equilibrio.

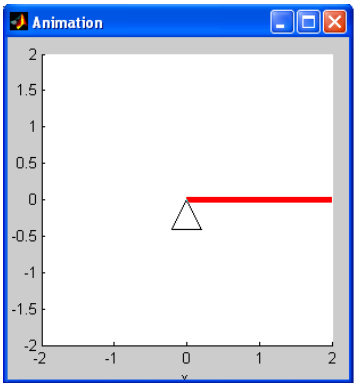


Fig. 6.10. Posición final del manipulador de dos enlaces.

Caso 5.- Para este ejemplo se tomará los mismos parámetros del caso anterior, sólo se cambiará la posición en la que se desea que el manipulador mantenga el equilibrio.

En DiBotMat	En PLANMANT
$DH = \begin{matrix} q(1) & 0 & 1 & 0 & theta1 \\ q(2) & 0 & 1 & 0 & theta2 \end{matrix}$	$L = [1 \quad 1]$
$Mr = \begin{matrix} -0.5 & 0 & 0 & 1 \\ -0.5 & 0 & 0 & 0 \end{matrix}$	$Lc = [0.5 \quad 0.5]$
$Ml = \begin{matrix} 0 & 0 & 0 & 0 \end{matrix}$	$m = [1 \quad 0]$
$g = \begin{matrix} 0 & -9.81 & 0 & 0 \end{matrix}$	$ml = [0]$
theta1=60°	$II = [0.25 \quad 0.25]$
varTheta1=0	$Bv = [0 \quad 0]$
vvarTheta1=0	$q0 = [pi/3; 0]$
	$qd0 = [0; 0]$

El par que se ha calculado en DiBotMat es:

$$Por\ Lagrange - Euler: \quad tl = \begin{matrix} 2.4525 \\ 0 \end{matrix} = \begin{matrix} 9.81 * 0.5 * \cos(pi/3) \\ 0 \end{matrix}$$

$$Por\ Newton - Euler: \quad tn = \begin{matrix} 2.4525 \\ 0 \end{matrix} = \begin{matrix} 9.81 * 0.5 * \cos(pi/3) \\ 0 \end{matrix}$$

Debido a que el enlace 2 no posee masa alguna entonces no hay un torque aplicado en él. En la fig. 6.11 se muestra que para este torque calculado el manipulador se mantiene en su posición inicial (60°; 0).

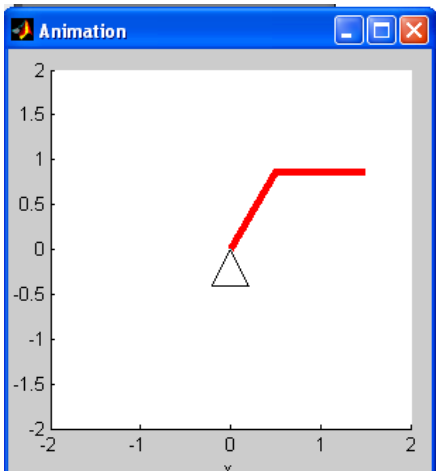


Fig. 6.11. Posición final del manipulador (60°; 0).

Las situaciones expuestas anteriormente se realizaron con la finalidad de comprobar la validez de los algoritmos computacionales desarrollados en este Trabajo de Grado a través del uso de la PLANMANT; aquí se plantea un sistema de simulación donde se estudia la dinámica directa de los manipuladores definidos por el usuario, donde el torque de entrada es calculado a través de DiBotMat. En definitiva, se logró comprobar que el torque calculado es el correcto para alcanzar la condición de equilibrio en los brazos del robot en una posición previamente especificada.

CONCLUSIONES

Luego de haber finalizado con el desarrollo de la toolbox e Interfaz Gráfica de Usuario “DibotMat” se concluye:

- La Robótica es una ciencia que exige un conocimiento considerable de las diferentes herramientas matemáticas que son aprendidas durante la formación del estudiante de ingeniería. Herramientas como sistemas coordenados, álgebra matricial, transformaciones homogéneas fueron conocimientos que se pusieron en práctica en este estudio.

- Es indispensable la comprensión de las metodologías Lagrange-Euler y Newton-Euler para ejecutar de una manera ordenada cada uno de los pasos que involucran la realización del proyecto.

- El modelo dinámico inverso permite expresar las fuerzas y pares que intervienen en función de la evolución de las coordenadas articulares y sus derivadas.

- Para la definición de parámetros es necesario que el usuario tenga por lo menos conocimientos básicos de Robótica Industrial, debido a que los parámetros ayudan a proporcionar el modelo matemático para el estudio de los manipuladores.

- La comprobación del correcto funcionamiento de los *scripts* y las funciones para la aplicación en Matlab de los algoritmos computacionales correspondientes a las formulaciones de Lagrange-Euler y Newton-Euler, demuestra que la Toolbox arroja las ecuaciones del movimiento necesarias para la modelación dinámica de los manipuladores.

- En el desarrollo de los dos algoritmos computacionales se puede deducir que el de Lagrange-Euler posee poca eficiencia computacional pero genera las ecuaciones de una forma estructurada; en cambio, el de Newton-Euler, debido

a que está basado en operaciones matriciales, tiene mayor eficiencia pero sus ecuaciones las presenta de una manera poco estructurada.

- El uso de la Interfaz Gráfica de Usuario (GUI) para la modelación de algunos manipuladores, en los ejemplos desarrollados y su comparación con los resultados obtenidos (para casos estáticos) con la interfaz en Simulink de Leon Zlajpah (Planmant), muestra su validez y confiabilidad.

- DiBotMat, como es de fácil usabilidad y muy sencillo de interpretar, puede ayudar al estudiante de Robótica o al usuario de Matlab al estudio de los robots o al aprendizaje básico de los mismos debido a que posee los fundamentos teóricos y la ayuda necesaria para lograrlo.

RECOMENDACIONES

- Para el correcto funcionamiento de la Toolbox se debe introducir los parámetros como se especifica en capítulos anteriores y/o manual de usuario.

- Para una mejor comprobación de los resultados de los algoritmos computacionales se puede desarrollar en Simulink/matlab un diseño para la simulación propia de DiBotMat.

- Diseñar e implementar un robot planar basándose en los estudios realizados en este Trabajo de Grado.

- En la modelación dinámica de robots con más de 3 grados de libertad se recomienda el uso de computadoras con alta capacidad de procesamiento y más de 2 GB de memoria RAM.

- La toolbox no puede ser corrida en MATLAB de 64 bits en computadoras con Windows seven, hasta que se haga una nueva versión tomando en cuenta esta limitación.

- Como nuevo usuario en el uso de DiBotMat se sugiere abrir el documento adjunto en él para mayor comprensión y ayuda a la hora de la exploración.

REFERENCIAS BIBLIOGRAFICAS

[1] **Robots Industriales: Definición y Clasificación** [Documento en línea]. Disponible: http://cfievalladolid2.net/tecno/cyr_01/robotica/industrial.htm [consulta: Abril, 2010].

[2] **Robótica Industrial** [Documento en línea] Disponible: http://html.rincondelvago.com/robotica-industrial.html [consulta: Abril, 2010].

[3] BARRIENTOS, A., PEÑÍN, L., BALAGUER, C. y ARACIL, R., **Fundamentos de la Robótica,** 2da Edición, Editorial McGraw Hill, año 1997.

[4] **Estructura de un robot** [Documento en línea]. Disponible: http://cmapspublic2.ihmc.us/rid=1H2B5THKD51F5C1J48/morfologia%20de%20un%20robot.pdf [Consulta: Abril, 2010].

[5] **Monografías, Estación Robótica para ensamble** [Documento en línea]. Disponible: http://www.monografias.com/trabajos16/estacion-robotica/estacion-robotica.shtml [Consulta: Abril, 2010].

[6] **Coordenadas Homogéneas** [Documento en línea]. Disponible: robotica.li2.uchile.cl/el710/Clase2.pp [Consulta: Mayo, 2010].

[7] **Icaro.eii.us.es** [Documento en línea]. Disponible: http://icaro.eii.us.es/descargas/tema_4_parte_2.pdf [Consulta: Mayo, 2010].

[8] **Cinemática Directa- La guía de Matemática** [Documento en línea]. Disponible: http://matematica.laguia2000.com/general/cinematica-directa [consulta: Mayo, 2010].

[9] SANZ, W. (2009). **Cinemática de robots industriales**. Universidad de Carabobo. Dirección de Medios y Publicaciones, 1ra Edición.

[10] SANZ, W. (2003). **PGIBOTMEDIA**. [CD ROM]. Disponible en la Biblioteca Rental de la Escuela de Ingeniería Eléctrica de la Universidad de Carabobo.

[11] **Definición de centro de masa y de gravedad** [Documento en línea]. Disponible: http://html.rincondelvago.com/centro-de-masa-centro-de-gravedad-y-centroide.html [Consulta: Mayo, 2010].

[12] **Wikipedia Enciclopedia libre, Dinámica** [Documento en línea]. Disponible: http://es.wikipedia.org/wiki/Dinamica [consulta: Mayo, 2010].

[13] **Modelo dinámico** [Documento en línea]. Disponible: http://148.202.148.5/cursos/cc321/fundamentos/unidad3/tema3_4_2.html [consulta: Mayo, 2010].

[14] **Dinámica** [Documento en línea]. Disponible: http://proton.ucting.udg.mx/robotica/r166/r97/r97.htm [consulta: Mayo, 2010].

[15] **MODELADO DIMAMICO** [Documento en línea]. Disponible: http://moddin.pdf/doc/related-resources/1584342632/452/2155.pdf [Consulta: Octubre 2010]

[16] **Modelamiento, simulación y control dinámico predictivo** [Documento en línea]. Disponible: http://manglar.uninorte.edu.co/bitstream/10584/112/1/72333756.pdf [Consulta: Mayo, 2010]

[17] **Monografías, Introducción al MATLAB** [Documento en línea]. Disponible: http://www.monografias.com/trabajos5/matlab/matlab.shtml [consulta: Abril, 2010].

[18] **MATLAB (PDF)** [Documento en línea]. Disponible: http://bosque.udec.cl/~sram/manuals/mitutorial.pdf [Consulta: Mayo, 2010].

[19] **Breve introducción de aspectos generales de Matlab** [Documento en línea]. Disponible: http://personal.us.es/contreras/practica1.pdf [Consulta: Mayo, 2010].

[20] **MATLAB Tutorial** [Documento en línea]. Disponible: http://mit.ocw.universia.net/18.06/f02/related-resources/matlab.pdf [consulta: Mayo2010].

[21] Manual de GUI en Matlab parte I. [Documento en línea]. Disponible: http://es.scribd.com/doc/15532859/MANUAL-DE-GUI-EN-MATLAB [Consulta: Junio, 2010]

[22] **Upel** [1998]**. Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales**, Caracas.

[23] ZORRILLA, S. [1993]**. INTRODUCCIÓN A LA METODOLOGÍA DE LA INVESTIGACIÓN.**

[24] ZLAJPAH, LEON [2000]. **PLANAR MANIPULATORS TOOLBOX**, for use with Matlab/simulink.

[25] GONZALEZ, N y MONTAÑEZ A. [2005], **Diseño de algoritmo de control basado en modo deslizante aplicado a un brazo manipulador de dos grados de libertad**. Trabajo de Grado. Universidad de Carabobo.

APÉNDICES

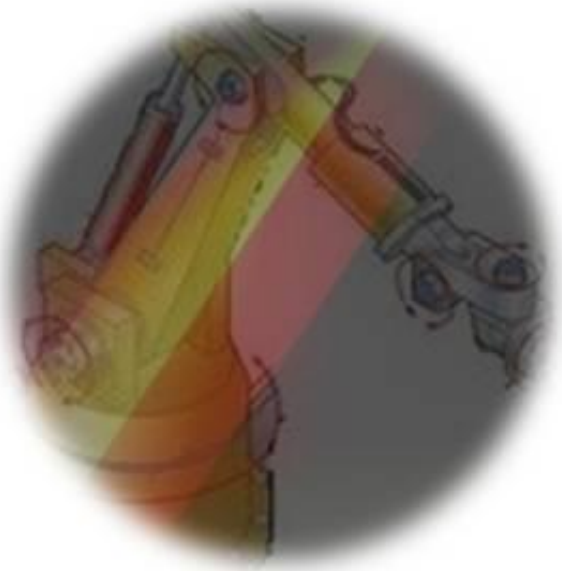
MANUAL DE USUARIO

DiBotMat

Modelación Dinámica de Robots Industriales

Br. Marilyn Calzada

Br. Milagros Ibarra



Tutor

Prof. Wilmer Sanz

El objetivo fundamental del manual es brindar al usuario de DiBotMat una ayuda en forma de guía, que le permita involucrarse con la toolbox e interfaz gráfica de usuario con mayor simplicidad y de manera sencilla, ofreciendo detalladamente explicaciones y ejemplos donde se visualice su funcionamiento. Su diseño es para ser comprendido por cualquier persona que esté al tanto del uso un computador, sin embargo, está dirigido a estudiantes de Ingeniería Eléctrica, Electrónica o Mecánica, de Especialización, Maestría o Doctorado que incluyan en sus plan de estudios cursos de Robótica y Mecatrónica.

Es muy importante tener en cuenta que la toolbox e interfaz gráfica de usuario "**DiBotMat**" se realizó en Matlab R2008a.

Uso de la Toolbox en Matlab

La toolbox en Matlab cuenta con varias funciones y dos scripts, dependiendo del algoritmo seleccionado se ejecuta el script adecuado para obtener el resultado esperado; Los dos (2) scripts corresponden con los algoritmos computacionales de Lagrange-Euler y Newton-Euler.

Los scripts en Matlab deben arrojar la opción de trabajar tanto con variables numéricas, y dar un resultado directo de los valores de las articulaciones; como con variables simbólicas asignando la opción de trabajar una ecuación característica que proporcione un modelo general de la dinámica del robot.

Para hacer uso de DiBotMat es necesario encontrarse en el entorno de trabajo de Matlab; situarse en el current Directory y se debe estar ubicado en DiBotMat. (Se sigue la dirección donde se encuentre guardada la toolbox DiBotMat, es decisión del usuario donde guardarla).

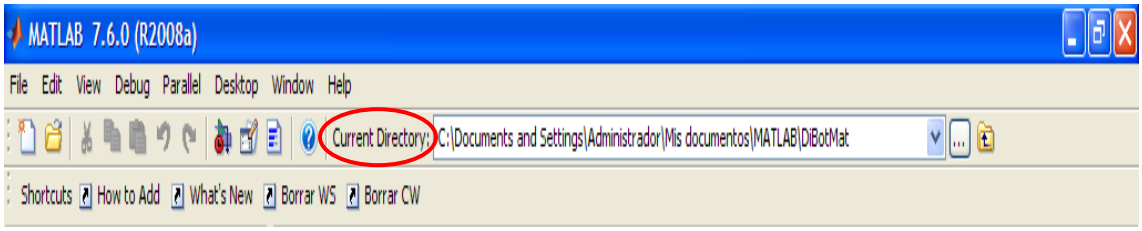


Fig. 1. Ubicación de la toolbox en Matlab.

Para el funcionamiento de la toolbox, se deben realizar los siguientes pasos en el espacio de trabajo de Matlab (command window) siguiendo el orden señalado:

1. Generar las variables articulares y sus derivadas necesarias para la formulación. De acuerdo al número de articulaciones que posea el robot, será el número de componentes de los vectores generados.
   q(1xn), varq(1xn), vvarq(1xn)
   ```
   [q varq vvarq]=genvar(ta)
   ```
   n: numero de articulaciones
   ta: vector que representa el tipo de articulación del robot y su longitud depende de n.

2. Definir los parámetros D-H. DH(nx4) '[theta1 d1 a1 alfa1;…]'
   ```
   DH=[q(1) d1 a alfa;theta2 q(2) a alfa;…]
   ```
   En donde las variables articulares; (`q(1), q(2)`) tienen tratamiento simbólico y el resto de los parámetros tratamiento numérico.

3. Definir la masa y los centros de masa de los elementos.
   Mr(nx4) '[xmr1 ymr1 zmr1 mr1;…]'
   ```
   Mr=[xmr1 ymr1 zmr1 mr1;….]
   ```

4. Definir la masa y los cetros de masa de la carga.
   Ml(1x4) '[xml1 yml1 zml1 ml1]'
   ```
   Ml=[xml1 yml1 zml1 ml1;….]
   ```

5. Generar la matriz M, la cual contiene los centros de masas y las masas de todo el sistema (robot+carga)

M(nx4) '[xm1 ym1 zm1 m1;…]'

```
M=com(Mr,Ml)
```

Para realizar el llamado de la función `com` es obligatorio que estén creados los argumentos de entradas requeridos `(Mr,Ml)`. Por eso es importante seguir el orden expuesto para el funcionamiento de la toolbox.

6. Definir la gravedad respecto al sistema de la base s0.

g(1x4) '[gx gy gz 0]'

```
g=[gx gy gz 0]
```

7. En este paso se realiza el llamado al script adecuado, de acuerdo al algoritmo computacional para el cual desee realizar el modelo dinámico del robot. En este caso se realizara primero el llamado al script correspondiente con la formulación de Lagrange-Euler (paso 7.1) y luego se realizará para Newton-Euler (paso 7.2), pero es indistinto cual de las dos se realiza primero o si simplemente se hace el llamado de una de las dos formulaciones antes mencionadas.

7.1. Llamado de Algoritmo de Lagrange-Euler (`lageu`) para la obtención del modelo dinámico del robot.

7.2. Llamado de Algoritmo de Newton-Euler para la obtención del modelo dinámico del robot. (`nweu`)

8. Introducir condiciones iníciales de las variables articulares. (Este paso se pudo haber realizado en cualquier momento antes del paso 7, sin ocasionar alteración alguna)

Si la primera variable articular es de tipo rotacional, será:

`theta1=θ; varTheta1=v; vvarTheta1=a;`

En caso de que la primera variable articular sea prismática:

`d1=d; varD1=v; vvarD1=a;`

Y así sucesivamente se van definiendo las condiciones iníciales de acuerdo al número de articulaciones que posea el robot. `θ, v, a, d,` son constantes numéricas.

9. Se evalúa cada una de las ecuaciones de torques y fuerzas arrojadas por los scripts, con las condiciones iníciales introducidas anteriormente con el fin de obtener resultados numéricos. Se realiza este paso a través de la función `eval.`

Ejemplo:

Dado el robot polar de dos grados de libertad (θ1, d2), Aplicar el método de Lagrange-Euler para la modelación dinámica de este manipulador.

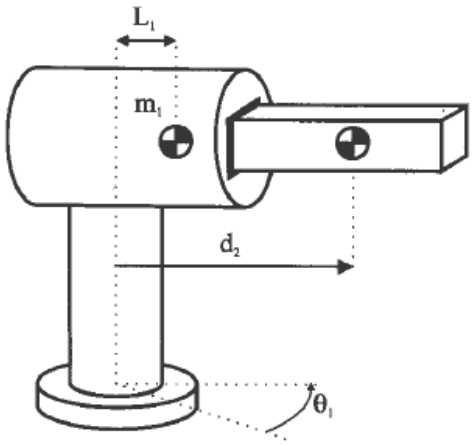


Fig. 2. Robot polar con dos articulaciones.

Tomando en cuenta los sistemas de referencia, se establecen los parámetros que se necesitan.

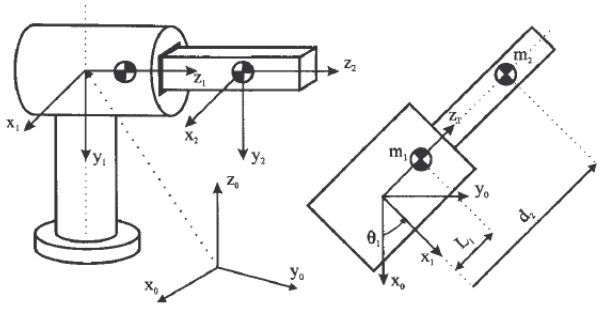


Fig. 3. Sistemas de referencia del robot polar.

Parámetros Denavit-Hartenberg:

Articulación	Θ	d	a	A
1	θ1	0	0	-90
2	0	d2	0	0

Centros de Masa y Masa del robot:

Articulación	X	Y	Z	Masa
1	0	0	1	1
2	0	0	0	1

Centro de Gravedad

X	Y	Z
0	0	-10

Ahora se procede a aplicar los pasos ya conocidos al ejemplo, para el funcionamiento de la toolbox, para eso se puede editar el siguiente código en el espacio de trabajo de Matlab:

```
>> [q varq vvarq]=genvar(['r' 'p']);
>> DH=[q(1) 0 0 -90; 0 q(2) 0 0];
>> Mr=[0 0 1 1; 0 0 0 1];
>> Ml=[0 0 0 0];
>> M=com(Mr,Ml);
>> g=[0 0 -10 0];
```

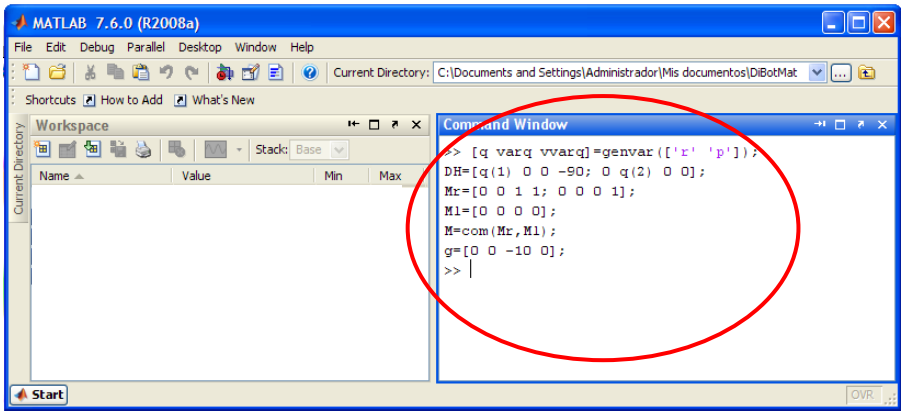


Fig. 4. Parámetros editados en el command window.

Cada una de estas líneas de código corresponde con los pasos del 1 al 6 respectivamente. Al introducir en el command window, este código se observa en la fig. 5. Que fueron creadas las variables en el workspace de Matlab.

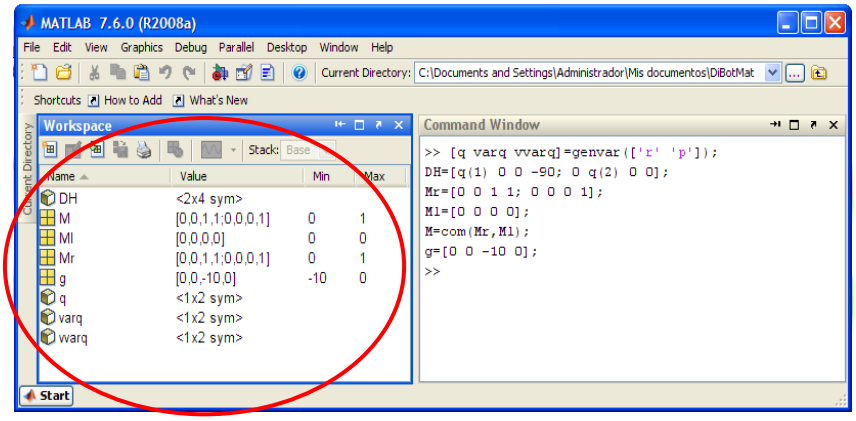


Fig. 5. Parámetros generados en el workspace.

Ahora se hace el llamado al script, de acuerdo al modelo que se requiere para efectos del manual, se hará el llamado de ambos, este paso 7 se puede editar en el commnd window: `>> lageu`

Se observa que se genera la ecuación dinámica del sistema mediante la formulación de Lagrange-Euler.

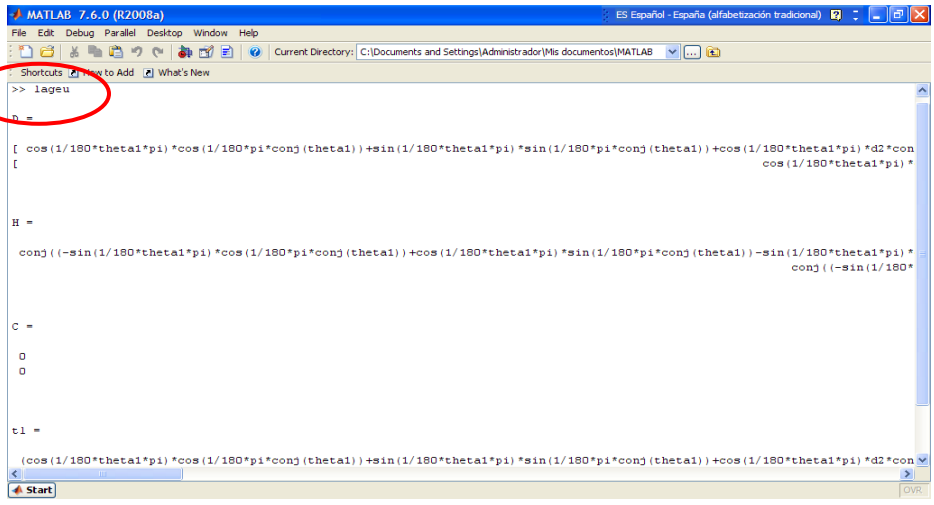


Fig. 6. Modelo dinámico a través del algoritmo de Lagrange-Euler.

Para la formulación de Newton-Euler se edita: `>> nweu`

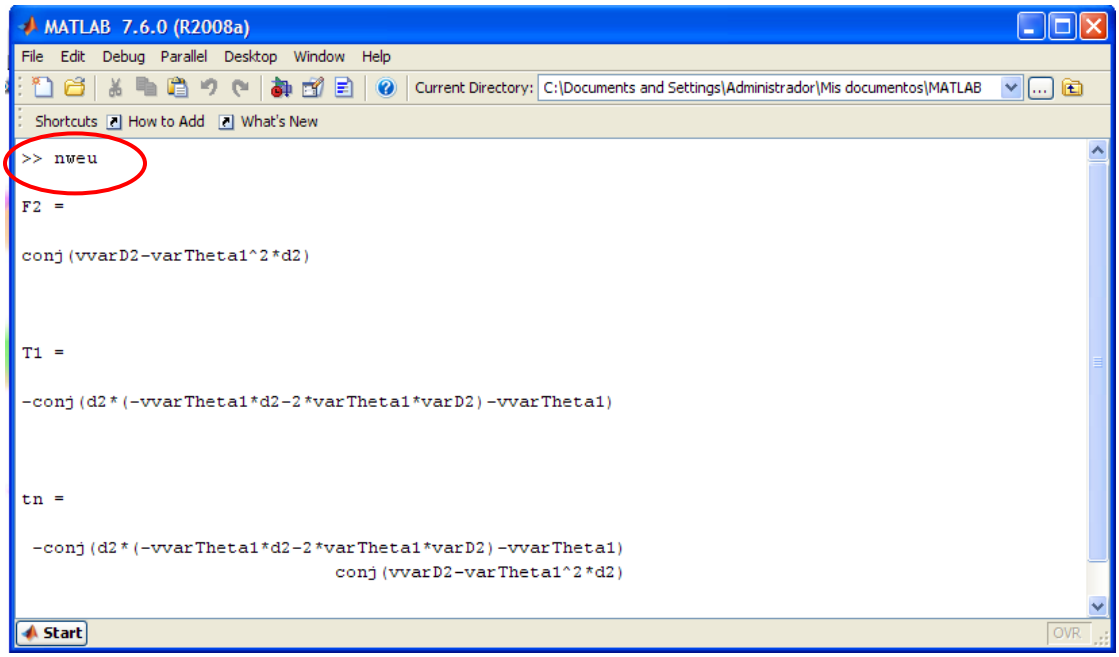


Fig. 7. Modelo dinámico a través del Algoritmo de Newton-Euler.

Se procede a introducir las condiciones iníciales como muestra el paso 8, para este ejemplo, suponga que la articulación 1 rotó 90° a una velocidad constante de 10, por lo que la aceleración es nula. Y la segunda articulación se desplazó 0.5 a una velocidad de 5 y aceleración nula.

Luego se obtiene el resultado numérico de las formulaciones evaluando las ecuaciones arrojadas por los scripts, como indica el paso 9. En el espacio de trabajo de Matlab se escribe lo siguiente:

```
>> theta1=90; varTheta1=10; vvarTheta1=0; d2=0.5; varD2=5; vvarD2=0;

>> tle=eval(tl)
```

```
>> tne=eval(tn)
```

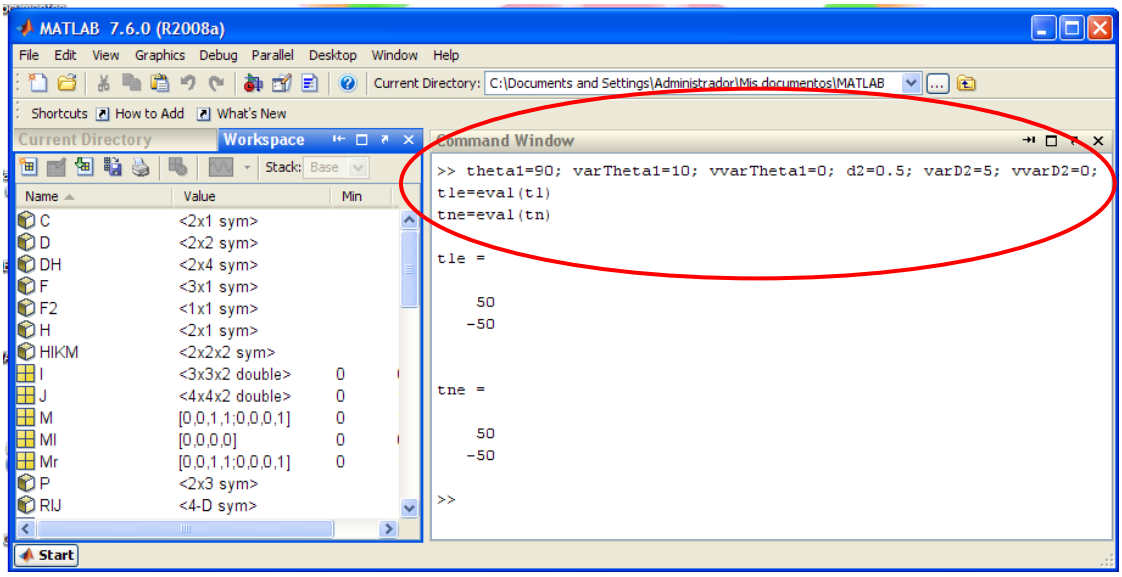


Fig. 8. .Valores de variables articulares y evaluación de formulaciones.

Uso de las funciones de la toolbox a través del ejemplo mostrado en la Fig. 2.

Función mph

Para el cálculo de mph, es obligatorio que los pasos 1 y 2, se encuentren creados y guardados en el workspace como muestra la fig.9.

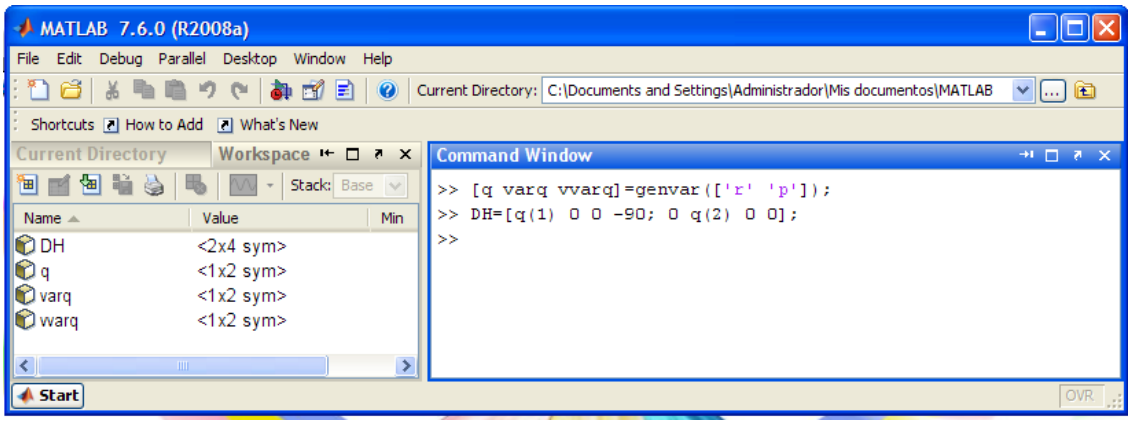


Fig. 9. Verificación de parámetros del workspace.

Y se procede con el llamado de la función editando el código:

```
>> A=mph(DH, 0, 1)

>> A=mph(DH, 1, 2)

>> A=mph(DH, 0, 2)
```

En la fig. 10. Vemos que la primera línea de código editada, arroja la matriz de transformación de la primera articulación. La segunda línea la mph de la segunda articulación, mientras que la tercera línea da la matriz de transformación del sistema completo.

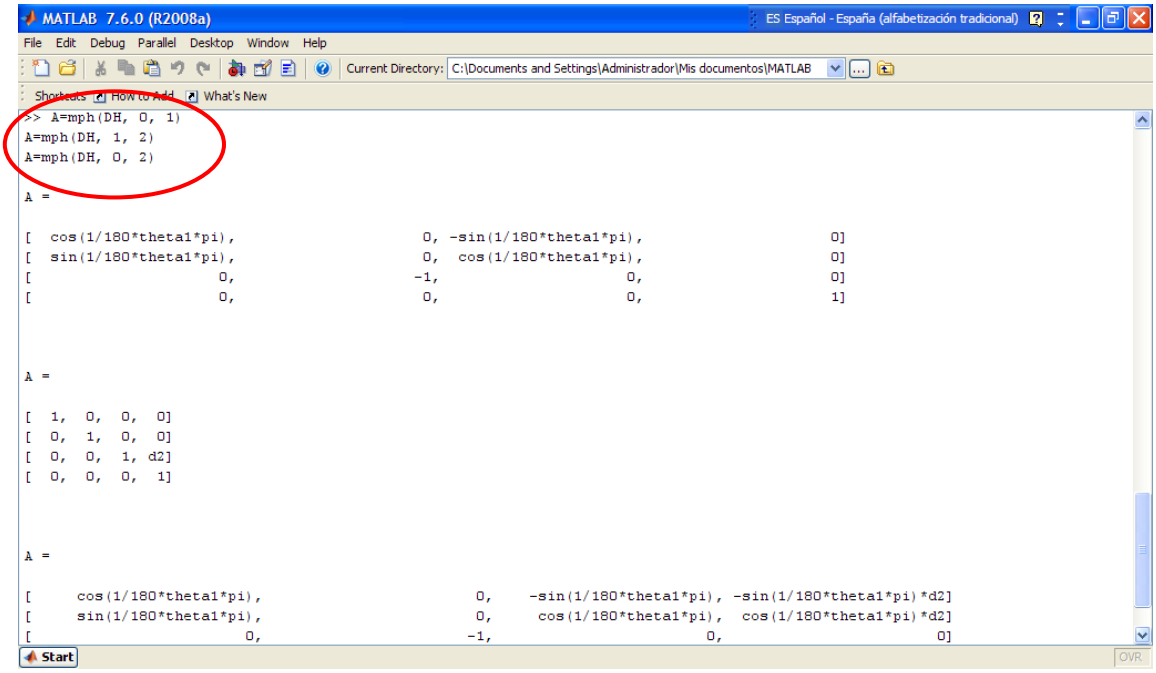


Fig. 10. Llamado de la función mph.

Si se desea visualizar la matriz de transformación numéricamente se introducen los valores numéricos de las variables articulares y se realiza la evaluación de la misma así como se muestra a continuación ejecutado:

```
>> theta1=90; d2=0.5;
```

```
>> eval(A)
```

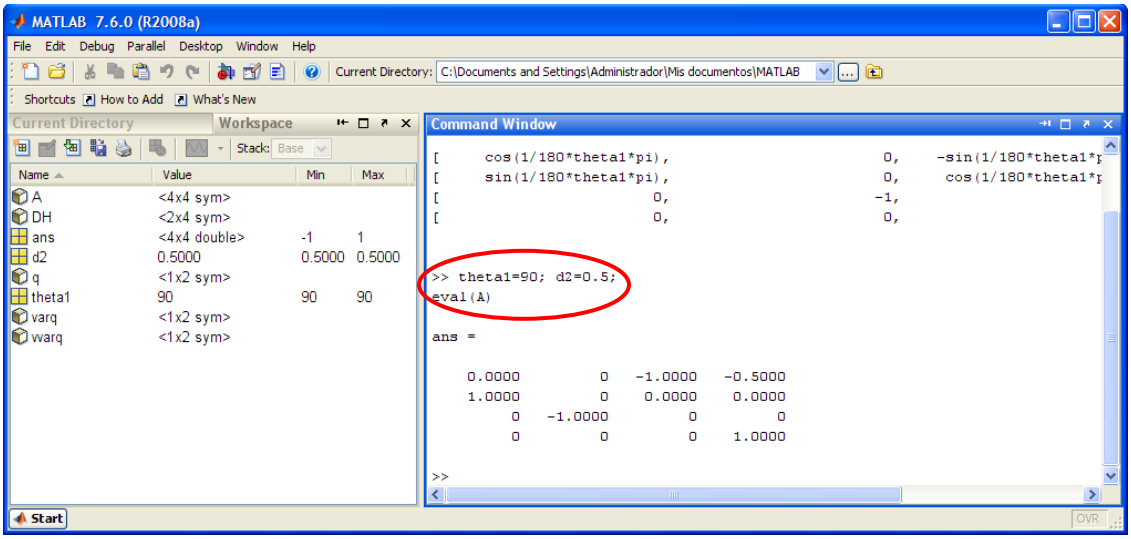


Fig. 11. Evaluación de la función mph.

Función uij

Uij necesita como argumentos de entrada DH y q, se verifica si están creados en el workspace, en caso contrario, se crean realizando los pasos concernientes, observe fig. 9. Se ejecuta el llamado de la función a través del siguiente código:

```
>> U=uij(DH,q)
```

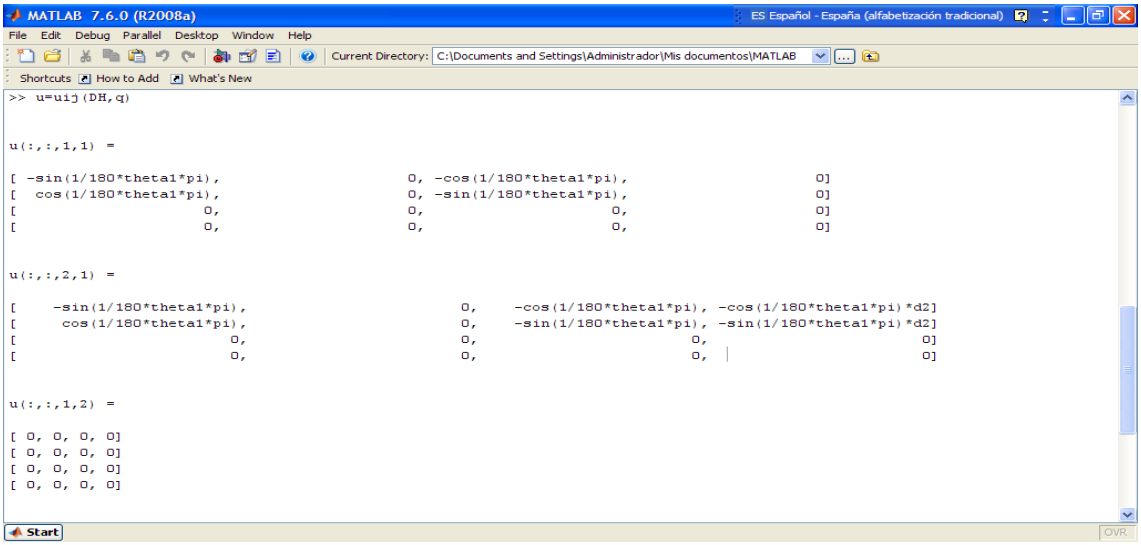


Fig. 12. Llamado de la función uij.

Escriba en Matlab, para observar resultados numéricos:

```
>> theta1=90; d2=0.5;

>> eval(U(:,:,2,2))
```

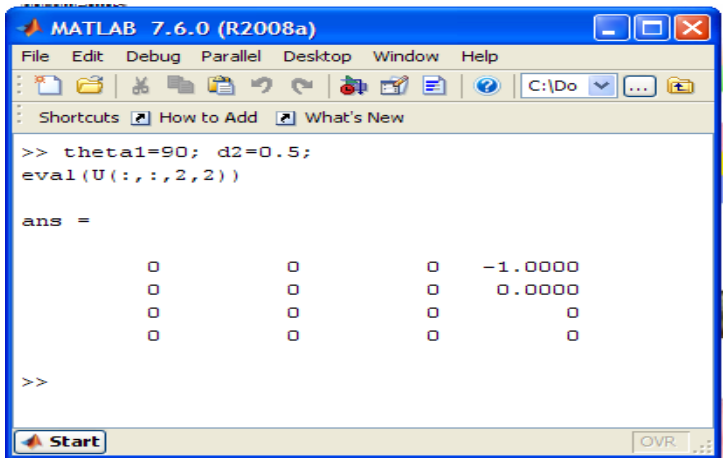


Fig. 13. Evaluación de la función uij.

Si desea ver numéricamente cada una de las matrices uij debe aplicar la función `eval` a cada una de ellas. Como se realizo en la fig. 13.

Función uijk

Visualizar fig. 9. En caso de no tener guardadas las variables de entrada de la función en el workspace para ejecutar el código:

```
>> UK=uijk(DH,q)
```

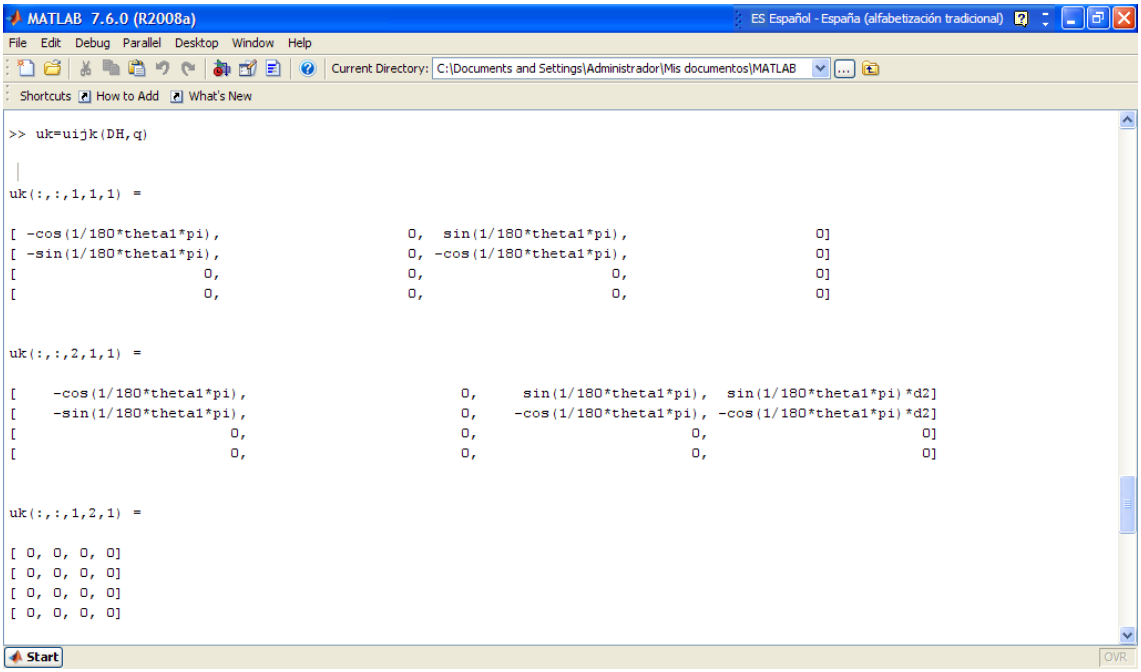


Fig. 14. Llamado de la función uijk.

Como se observa a través de la fig. 15. los valores numéricos de las variables articulares están creados en el workspace.

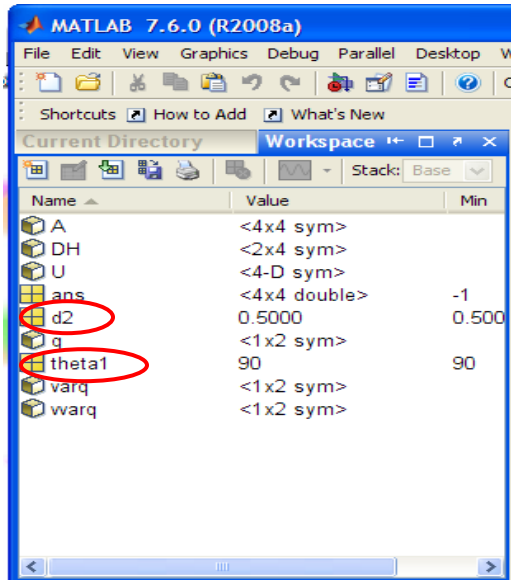


Fig. 15. Valores numéricos de variables articulares.

En este caso solo se hace ejecuta `eval` para obtener los resultados numéricos de cada una de las matrices. Solo para muestra al usuario se evalúa una sola matriz escogida arbitrariamente. Se puede verificar copiando en Matlab:

```
>> eval(UK(:,:,2,1,1))
```

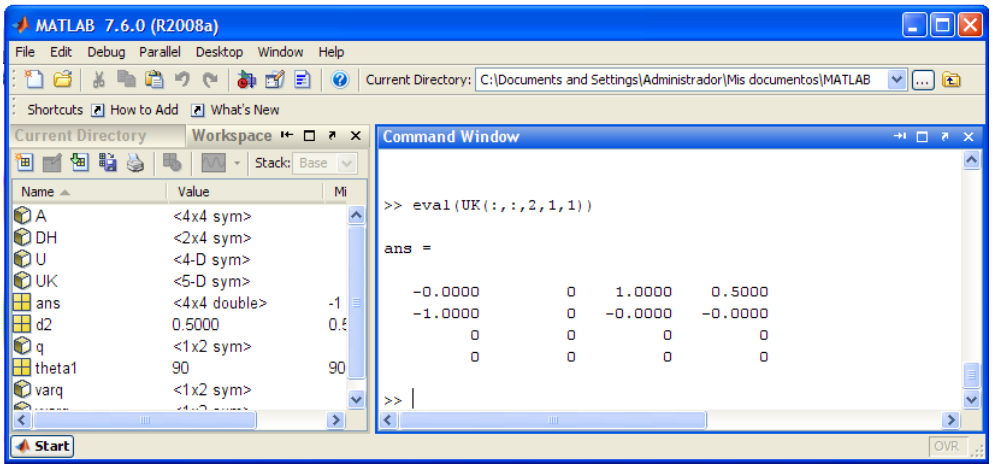


Fig. 16. Evaluación de la función uijk.

Función ji

Para el cálculo de Ji se requiere conocer la masa del sistema, si se encuentra creada en el workspace se hace el llamado directo de ji, sino se edita en el command window como muestra la figura:

```
>> Mr=[0 0 1 1; 0 0 0 1];

>> Ml=[0 0 0 0];

>> M=com(Mr,Ml);
```

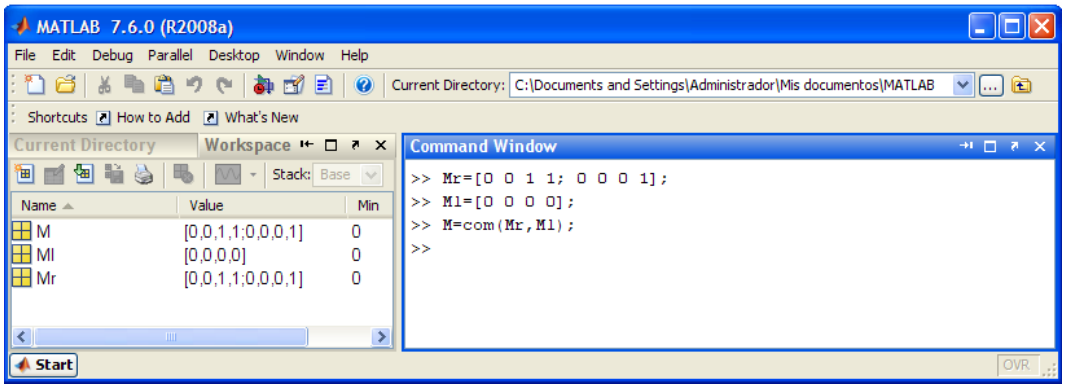


Fig. 17. Introducción de la masa del sistema.

Se procede a calcular ji editando:

```
>> J=ji(M)
```

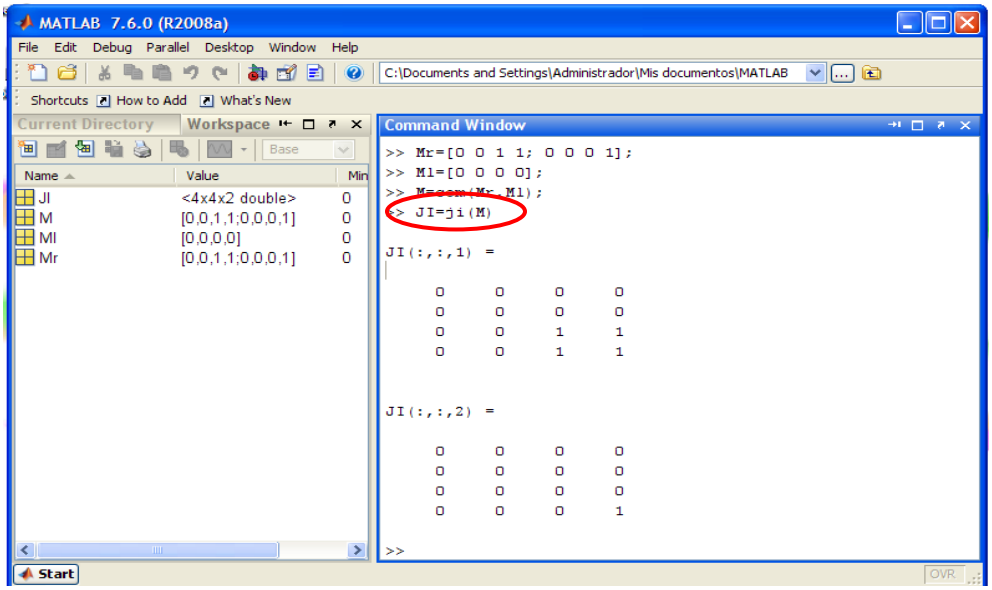


Fig. 18. Llamado de la función JI.

Función d

Una vez que se verifique todas las variables de entradas requeridas en el workspace:

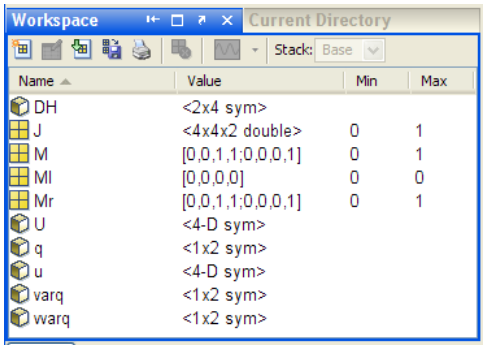


Fig. 19. Verificación de datos de entrada de D.

Se introduce en la ventana de comandos:

```
>> D=d(U,J)

>> theta1=90; d2=0.5;

>> eval(D)
```

Obteniendo el resultado de la función d, que se muestra la fig. 20.

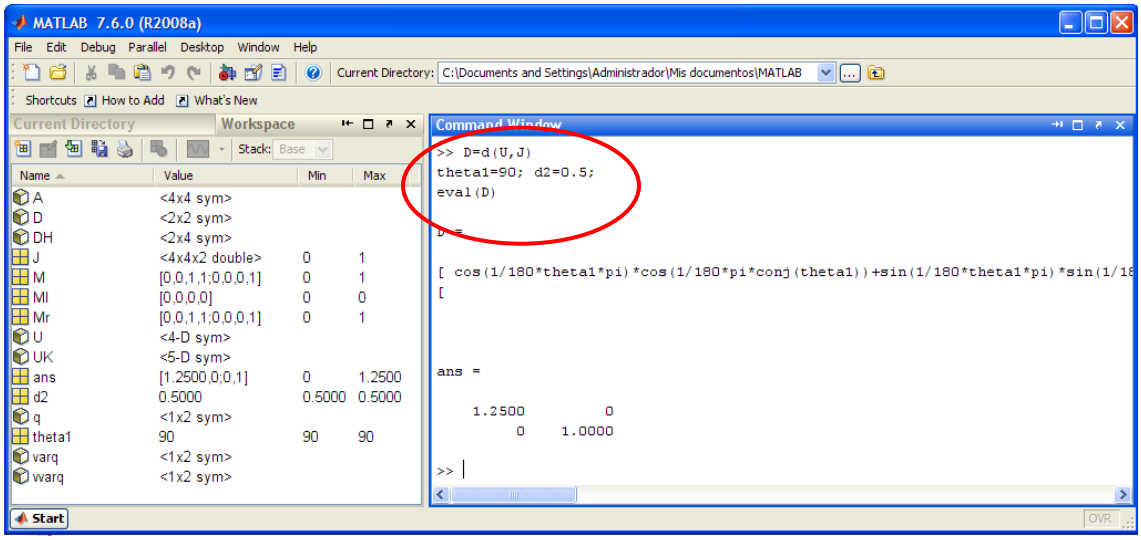


Fig. 20. Llamado y evaluación de la función D.

Función hikm

Asumiendo que se encuentran las variables de entrada de la función hikm en el workspace, ejecutamos:

```
>> HIKM=hikm(UK,U,J)

>> eval(HIKM(:,:,1))

>> eval(HIKM(:,:,2))
```

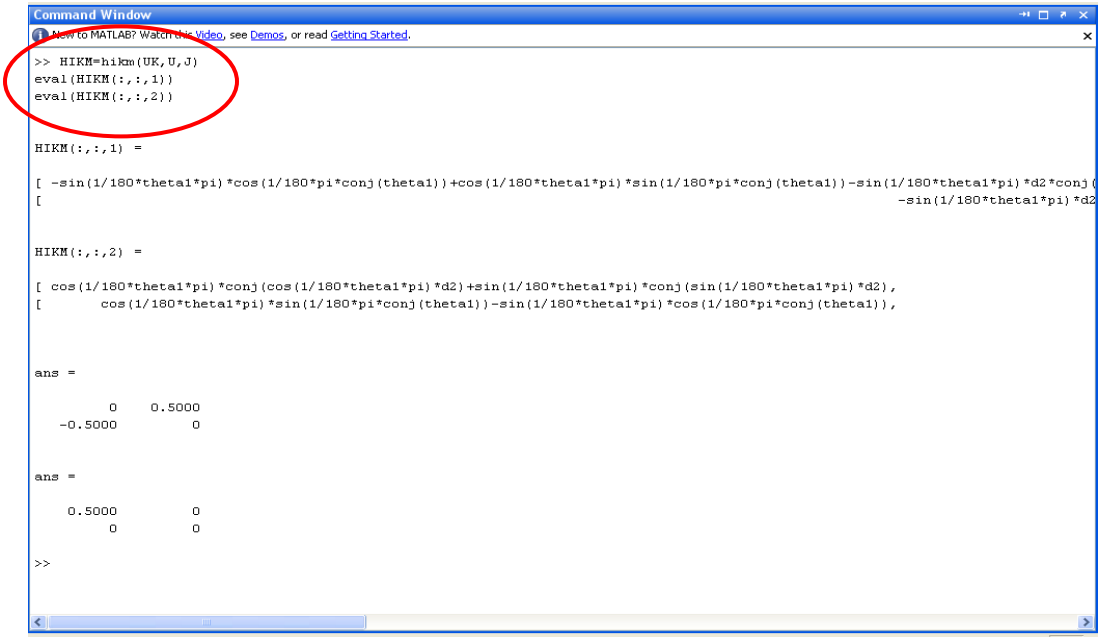


Fig. 21. Llamado y evaluación de la función hikm.

Función h

Considerando que ya fueron ejecutadas las variables de entrada de la función, se hace el llamado de la misma por medio de:

```
>> HI=h(HIKM,varq)
```

Luego, si se desea evaluar, se introducen los valores de las variables articulares como se muestra:

```
>> varTheta1=10; varD2=5;

>> eval(HI)
```

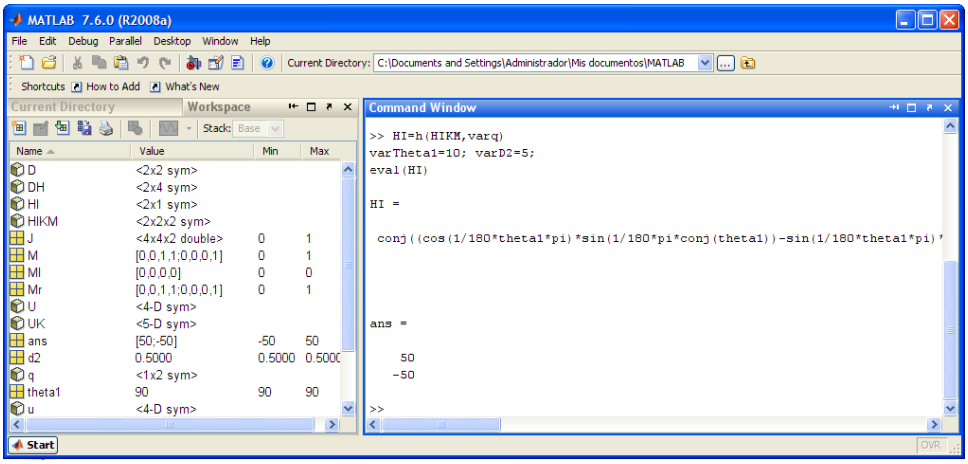


Fig. 22. Llamado y evaluación de la función h.

Función c

En el caso donde no se encuentre ninguna variable guardada en el workspace realizar los pasos del 1 al 6, revise fig. 4. y luego proceda a realizar los llamados ejecutando en Matlab:

```
>> U=uij(DH,q);

>> CI=c(DH,M,g,U)
```

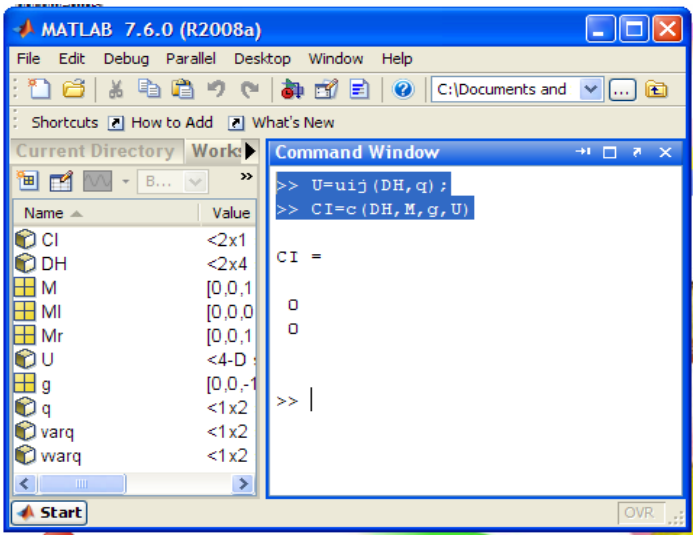


Fig. 23. Llamado de la función c.

Pues ahora el argumento de entrada de la función c, cambió, el nuevo vector gravedad g expresado en el sistema de la base es: g=(-10, 0, 0, 0), por tanto modificar en Matlab:

```
>> g=[-10 0 0 0];

>> CI=c(DH,M,g,U)
```

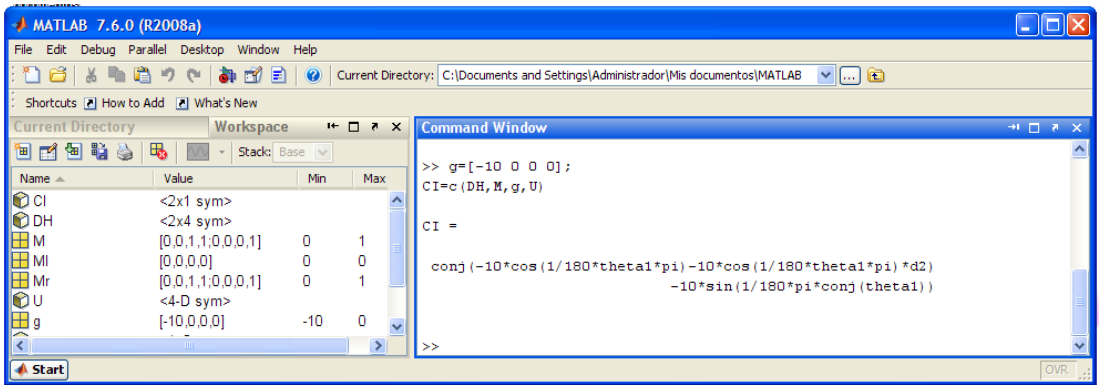


Fig. 24. Llamado de la función c cambiando g.

La función que representa las fuerzas de gravedad arrojó otro valor distinto, si el usuario requiere evaluarlo podrá escribir en Matlab:

```
>> theta1=90; d2=0.5;

>> eval(CI)
```

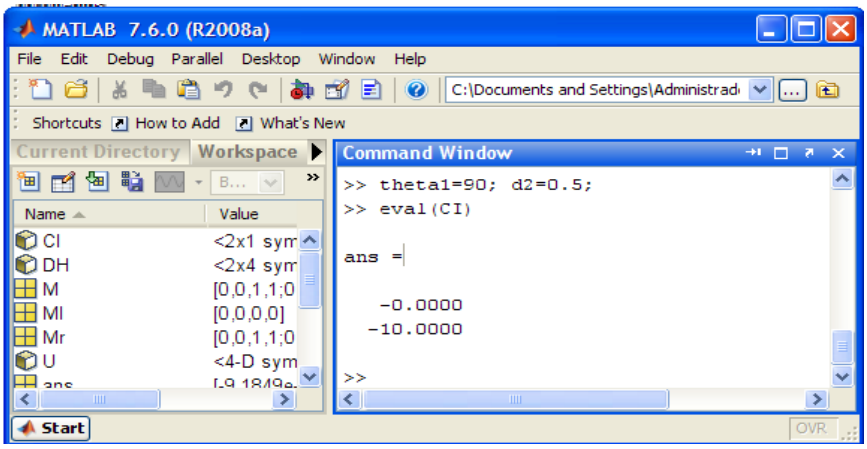


Fig. 25. Evaluación de la función c.

Función rot

Si editamos en el espacio de trabajo de Matlab el siguiente código:

```
>> ROT1=rot(DH, 0, 1)

>> ROT2=rot(DH, 1, 2)

>> ROT=rot(DH, 0, 2)

>> ROT=eval(ROT)
```

Es debido a que los argumentos de entrada están creados en el workspace, como se ve en la fig. 9.

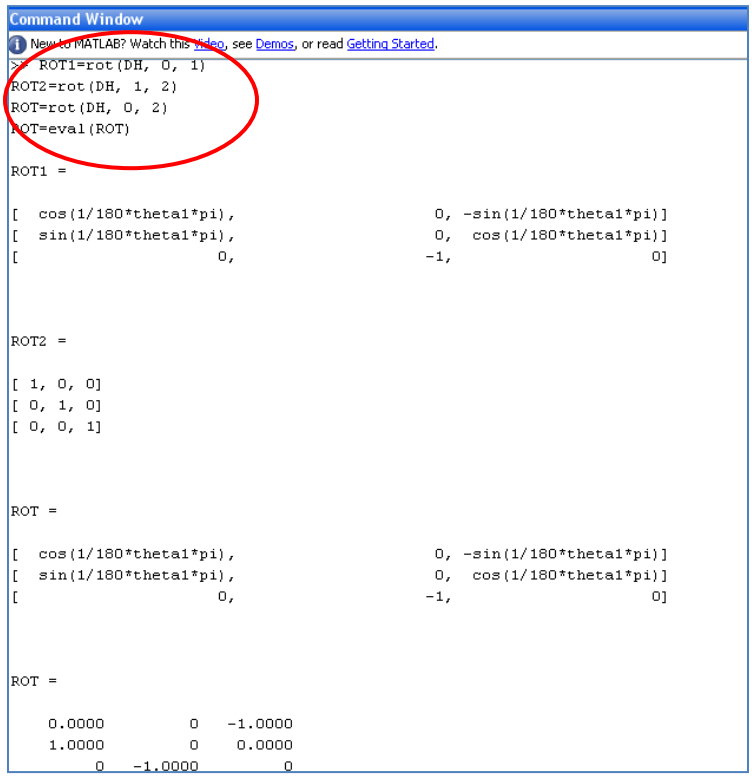


Fig. 26. Llamado y evaluación de la función rot.

Uso de la Interfaz Grafica de Usuario (GUI)

Para iniciar la GUI se bebe ejecutar la siguiente instrucción en la ventana del command window:

```
>> DiBotMat
```

Se presenta la siguiente ventana:

Herramienta para abrir el Manual de Usuario

Fig. 27. Presentación de DiBotMat.

En la fig. 27 se observa en la parte superior una barra que contiene la opción Abrir; al hacer click podrá abrir este manual y visualizarlo a medida que navega por la GUI; Fundamentos teóricos para la modelación dinámica así como una ayuda básica, esta barra se encuentra en todas las GUI-s principales de DiBotMat para facilitar su uso. Al hacer click sobre Fundamentos Teóricos se despliega el siguiente menú:

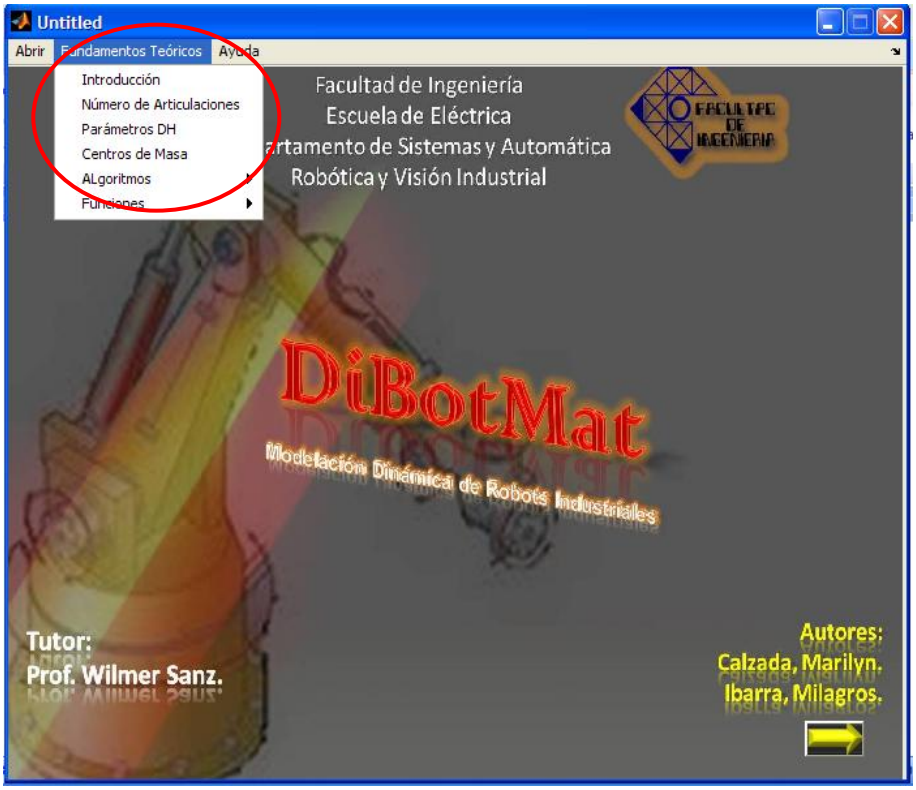


Fig. 28. Menú de Fundamentos Teóricos.

Para acceder al menú observado a través de la fig. 28, simplemente hacer click sobre cualquiera de las opciones mostradas, en caso que el usuario desconozca en qué basarse para introducir cualquier parámetro o necesite una breve refrescamiento acerca de la modelación dinámica por los métodos desarrollados de Lagrange-Euler y Newton-Euler.

Al hacer click sobre introducción en el menú de fundamentos teóricos de la fig. 28, se presenta la próxima GUI:

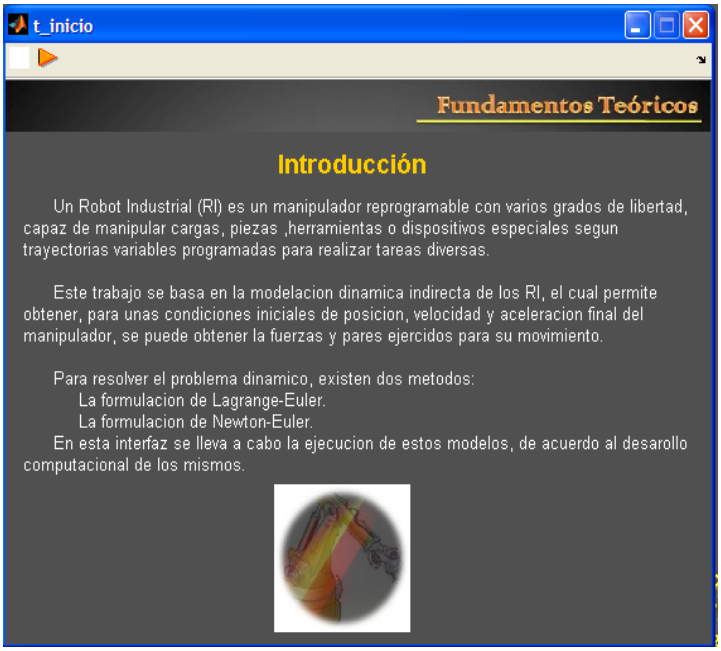


Fig. 29. Introducción de Fundamentos Teóricos.

Si se desea navegar y explorar todo el contenido de fundamentos teóricos puede desplazarse con facilidad presionando la flecha de la barra superior de la fig. 29, que abrirá la siguiente ventana y así moverse por todo el menú de manera sencilla.

Otra forma de conocer los fundamentos teóricos sin necesidad de entrar a su menú, es a través del botón que contiene el signo de interrogación, como lo muestra repetidas veces la fig. 31.

Si se hace click sobre Ayuda>>AYUDA de la barra de la fig. 27 se abrirá:

Fig. 30. Menú de Ayuda.

En la fig. 30 se observa del lado izquierdo de la GUI, los nombres de las ventanas principales, haciendo click en cada uno de ellos, se visualiza del lado derecho una guía básica acerca del uso de dichas ventanas.

Haciendo click sobre la flecha que se encuentra en la parte inferior derecha que indica continuar de la fig. 27, se va a la siguiente GUI que lleva por nombre *robotgui*. **Antes de continuar, para explicar el funcionamiento de la interfaz, se tomará el mismo ejemplo citado para el uso de la toolbox. Se va a obtener aquí el modelo dinámico del mismo robot de la fig. 2.**

Ahora se procede a realizar la modelación, utilizando DiBotMat.

Si tiene abierta la ventana mostrada en la fig. 31, es porque se encuentra en la segunda interfaz llamada *robotgui* y es allí donde se introducen los parámetros establecidos anteriormente, empezando por el número de articulaciones como se observa en la fig.32.

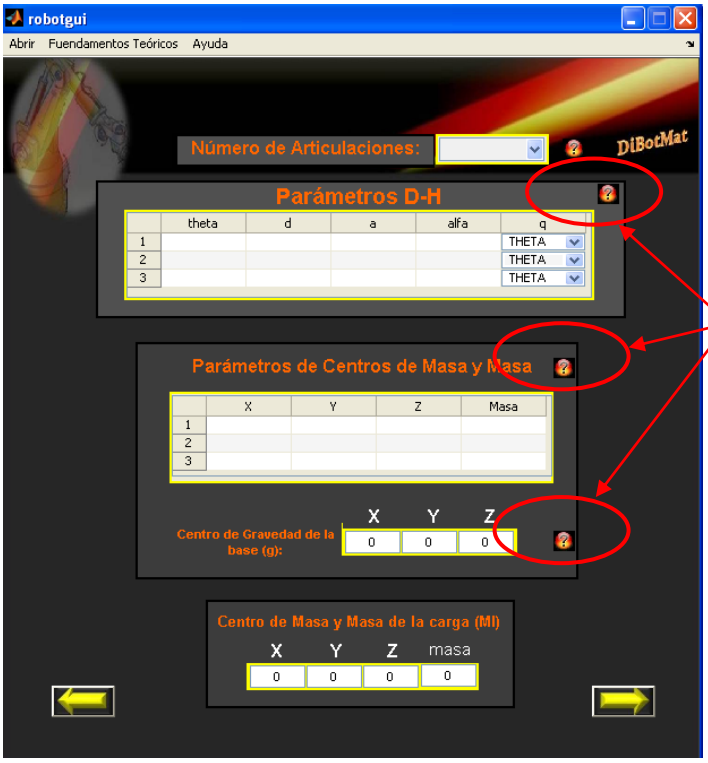


Fig. 31. Ventana *robotgui*.

Fig. 32. Numero de articulaciones del robot.

Al hacer click en dos, ver fig. 32 se configuran el tamaño de las tablas de abajo, quedando la interfaz de *robotgui* como se muestra en la fig. 33.

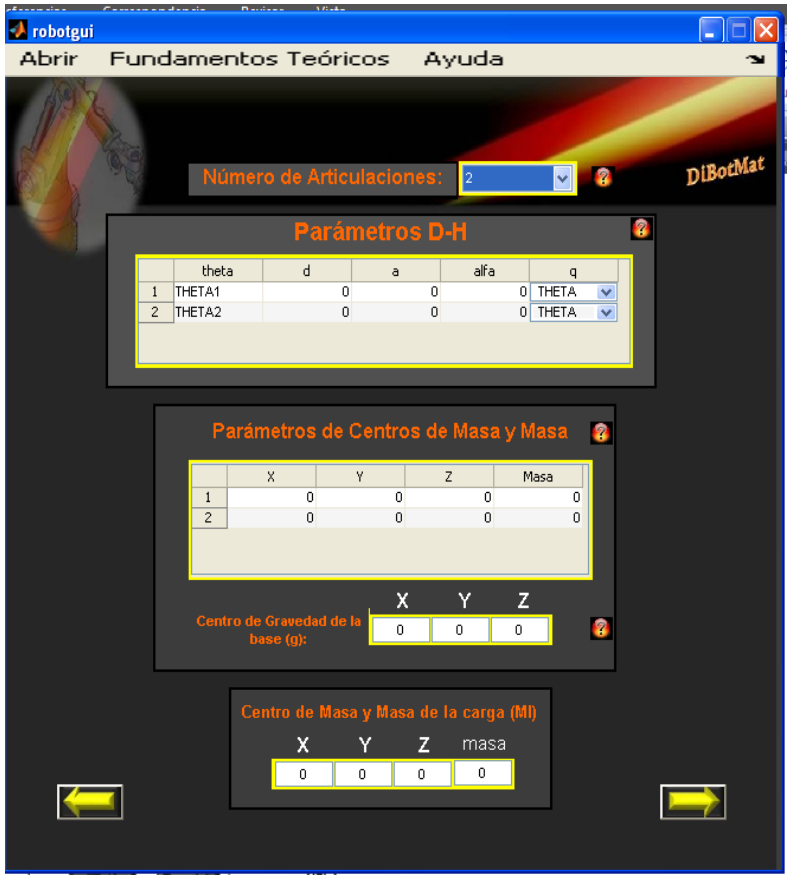


Fig. 33. Configuración de las tablas de los parámetros.

Ahora se introducen los parámetros D-H, empezando por la última columna llamada q, aquí se escoge si la articulación es prismática o rotacional, para efectos de este ejemplo se sabe que la primera articulación es rotacional y la segunda prismática. Fíjese en fig. 34 y fig. 35.

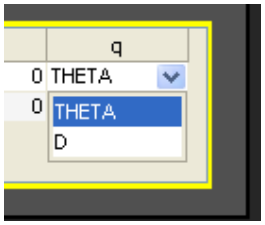


Fig. 34. 1º Variable articular.

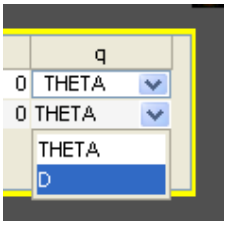
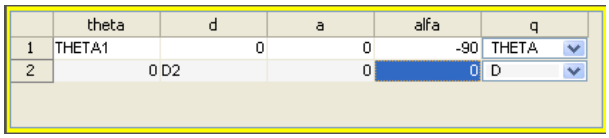


Fig. 35. 2º Variable articular.

Esto genera que la variable theta en la articulación 1, tome el valor simbólico de theta1 y la variable d en la articulación 2 tome el valor simbólico d2; ver fig. 36. Éstos no se modifican al introducir los demás parámetros de la tabla, los cuales se meten posicionándose en la celda correspondiente; y se DEBE dar **enter** para que se guarde el valor que escribió.

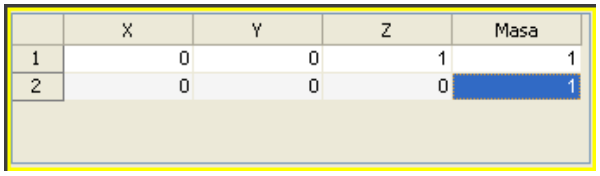
La tabla con los parámetros queda de la siguiente forma:



	theta	d	a	alfa	q
1	THETA1	0	0	-90	THETA
2	0 D2		0	0	D

Fig. 36. Tabla de parámetros DH.

Luego, se introduce los centros de masa y masa del robot, posicionándose en cada celda y después dar **enter** para que se guarde el valor escrito, esta tabla se visualiza en la fig.37.



	X	Y	Z	Masa
1	0	0	1	1
2	0	0	0	1

Fig. 37. Parámetros de centros de masa y masa del robot.

Para introducir el centro de gravedad, se posiciona en la casilla correspondiente a la coordenada Z, porque para este caso sólo posee valor en dicho eje; no es necesario dar **enter** luego de introducir el valor. Ver fig. 38.

Fig. 38. Gravedad de la base.

Ahora bien, como se muestra en la figura 2, en el efector final no existe ninguna carga sostenida, los últimos parámetros que son el centro de masa y la masa de dicha carga se puede dejar como está, ya que los valores por defecto que poseen todos es cero. Como en la fig. 39

Fig. 39. Centro de masa y masa de la carga.

Se presiona el botón que indica siguiente en *robotgui* y se pasa a la interfaz *modelogui*:

Fig. 40. Ventana *Modelogui*.

Como para este ejemplo se pide la modelación dinámica a través de ambos algoritmos, se presiona el botón correspondiente a cada formulación y se espera que aparezca el tiempo de ejecución, ver fig. 41. Lo que se logra con esto es generar las ecuaciones simbólicas en el command Windows y guardarlas en el workspace; luego se presiona el botón que indica siguiente y se abrirá la ventana *resultado* mostrado en la fig. 42.

Fig. 41. Algoritmos computacionales.

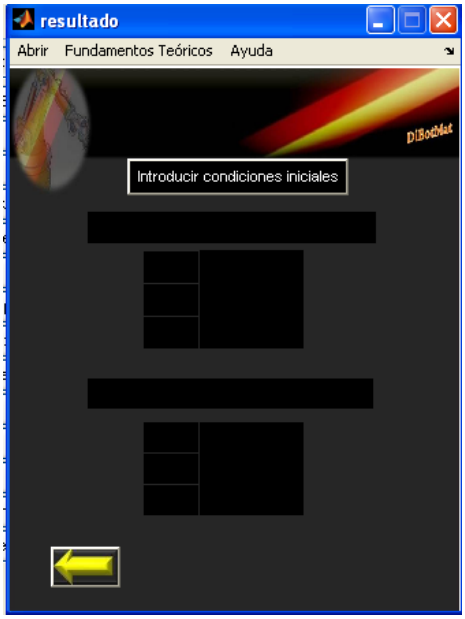


Fig. 42. Ventana *Resultado*.

Como se observa, esta ventana no muestra los resultados inmediatamente; primero se debe presionar el botón introducir condiciones iníciales (fig. 43), por lo que se abrirá la ventana *condinigui* que se observa en la fig. 44.

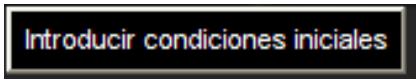


Fig. 43. Botón de condiciones iníciales.

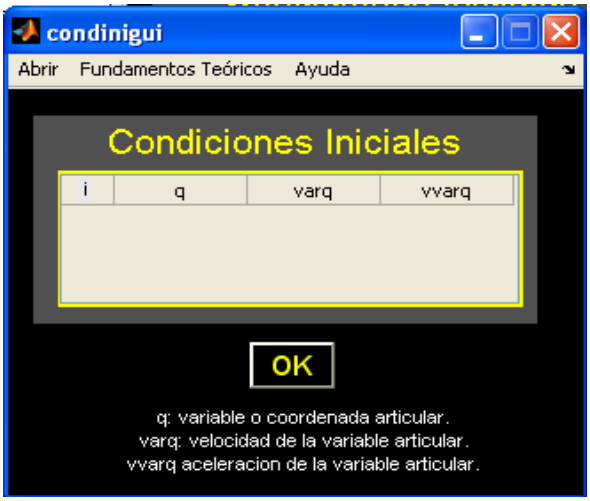


Fig. 44. Ventana *Condinigui*.

En *condinigui* se introducen las condiciones necesarias para conocer el torque o fuerza ejercida por cada articulación para moverse, del mismo modo que los parámetros del inicio.

¿Cuáles condiciones introducir?

Ejemplo

Suponga que la articulación 1 rotó 90° a una velocidad constante de 10°/s, por lo que la aceleración es nula. Y la segunda articulación se desplazó 0.5 m a una velocidad de 5 y aceleración nula. Estos valores introducidos en la tabla de *condinigui* se ven de la siguiente forma:

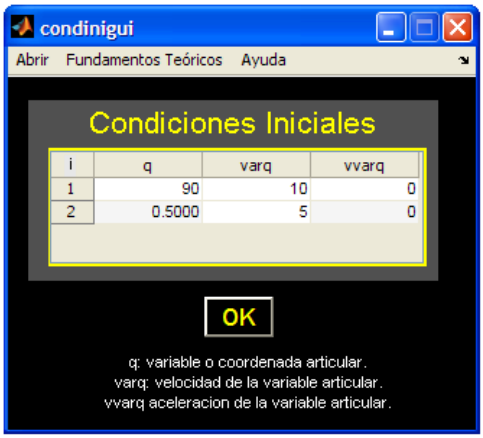


Fig. 45. Edición de datos.

No olvide **presionar enter** después de editar cada valor para ser guardado. Al presionar el botón OK de la fig. 45, se muestra en la ventana *resultado*, el modelo dinámico obtenido, se puede ver en la fig. 46.

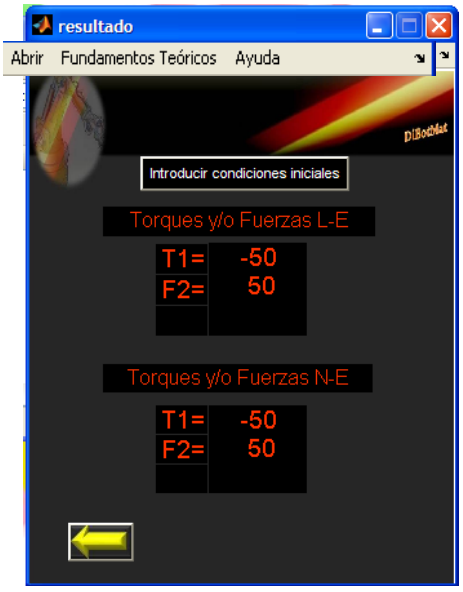


Fig. 46. Modelo dinámico del manipulador estudiado.

Si el usuario desea introducir otros valores numéricos basta con hacer click nuevamente sobre el botón de la fig. 43 para abrir nuevamente la ventana mostrada en la fig. 45 editar las nuevas condiciones y de esta manera conocer los nuevos resultados del modelo dinámico.

Ahora si el usuario requiere conocer alguna función de las utilizadas en los modelos de Lagrange-Euler y Newton-Euler, se debe situar en la ventana *modelogui* (fig. 40) y hacer click sobre es botón que muestra la fig. 47. Y se le presentara la interfaz mostrada en la fig. 48.

Fig. 47. Botón de crear función.

Fig. 48. Ventana *funciongui*.

Al presionar cualquier botón identificado con el nombre de la función que permite calcular, se abrirá la ventana donde se visualizarán los resultados de dicha función. Para el ejemplo de la fig. 2 se crearon algunas las funciones disponibles de esta ventana. Se comenzará por hacer click en el botón mph. Ver fig. 49 y se presentará la GUI observada en la fig. 50.

Fig. 49. Botón mph.

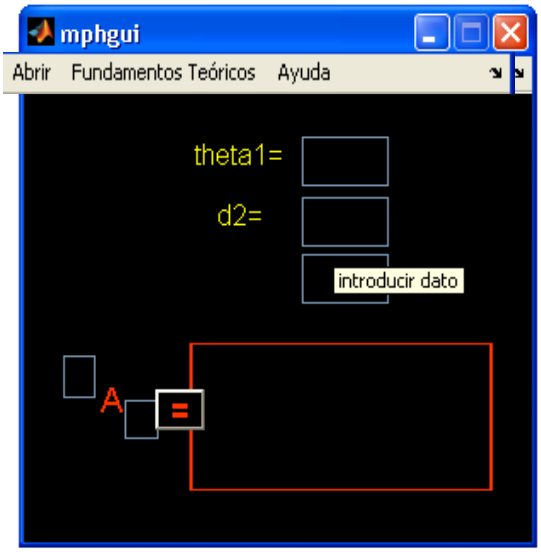


Fig. 50. Ventana *mphgui*.

Se introducen los valores de las variables articulares (resaltados en la fig. 51 en rojo) y se especifica para ver la matriz de transformación del eslabón i referido al eslabón i-1 (resaltados en la fig. 51 en azul) y se presiona el botón con el símbolo de igual para observar la matriz, vea fig. 52.

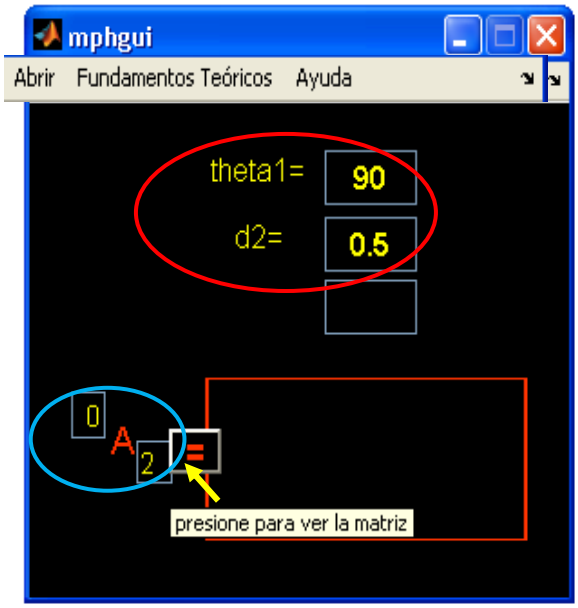


Fig. 51. Introducción de variables articulares.

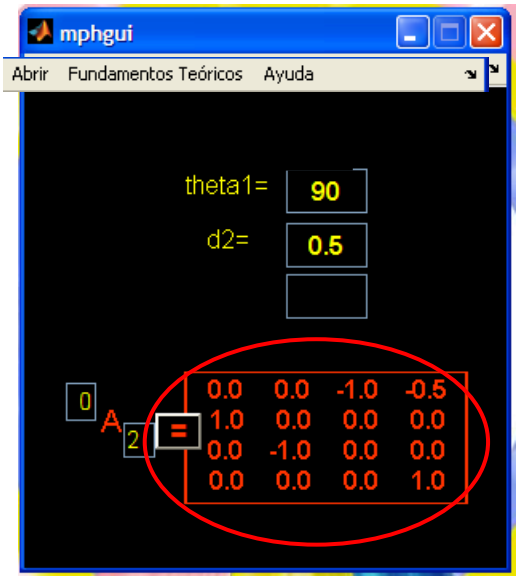


Fig. 52. Resultado numérico de la matriz mph.

Simplemente con editar los datos introducidos y presionar el botón de igualdad se puede conocer el nuevo resultado matriz, vea fig. 53 donde se resalta en color azul, el valor que se editó y el nuevo resultado de mph se observa en el cuadro de borde rojo.

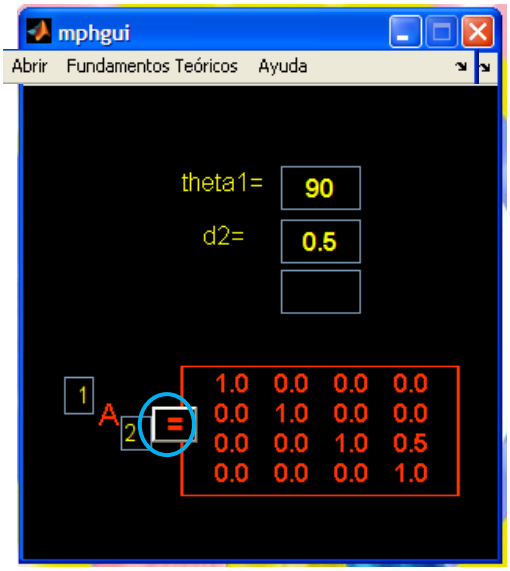


Fig. 53. Nuevo resultado matriz mph.

Como se observa en la fig. 52 y en la fig. 53 el resultado de la matriz fue impreso en el cuadro de borde rojo y los datos introducidos en los cuadros de borde azules, esto aplica para todas las GUIS-s de ventanas de resultados de las funciones.

Para el cálculo de la función ji se presiona el botón de la fig. 54 que se encuentra en la GUI de *funciongui* donde se presentará la ventana con los resultados impresos mostrados en la fig. 55, debido a que los datos para el cálculo de la matriz de pseudoinercia de cada elemento son la masa y la posición del centro de masas; y fueron introducidos en la segunda interfaz llamada robotgui (ver fig. 37).

Fig. 54. Botón Ji.

Fig. 55. Resultados matriz Ji

Para obtener la matriz de inercia a través de DiBotMat se hace click en el botón mostrado en la fig. 56 de la ventana *funciongui.*

Fig. 56. Botón D.

Se abrirá la ventana de la fig. 57 donde se debe aplicar el mismo proceso que se utilizó para obtener mph se introducen los datos pedidos por la GUI y se presiona el botón de igual para ver los resultados mostrados en la fig. 58.

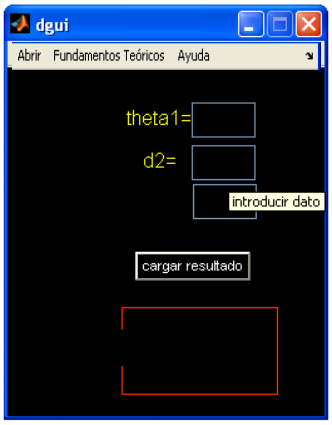


Fig. 57. Introducir datos de D.

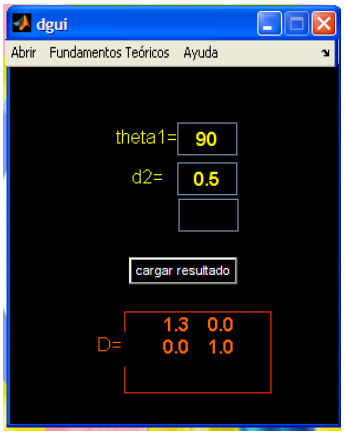


Fig. 58. Resultado matriz D.

APENDICE B

APENDICE B.1- Scripts y funciones de los Algoritmos Computacionales

APENDICE B.1.1

FUNCIÓN c.m

```
FUNCIÓN [CI]=c(DH,M,g,U,n)
%C   fuerzas de gravedad.
%    C=C(DH,M,G,U,N) Retorna la matriz de las fuerzas de gravedad. 'M' es
% la matriz de centros de masa, 'G' es el vector de gravedad expresado en
% el sistema de la base del robot temporales de las variables articulares
% y 'N' es el numero de grados de libertad.
%
%   Copyright (c) 2010 por TMM.
if nargin==4
    n=length(DH(:,1));
end
if nargin > 5
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 4
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
C=sym(zeros(1,n));
for i=1:n
    for j=1:n
        C(i)=C(i)-M(j,4)*g*U(:,:,j,i)*(mph(DH,j,j)*[M(j,1) M(j,2) M(j,3)
1]');
    end
end
CI=C';
```

APENDICE B.1.2

FUNCIÓN com.m

```
FUNCIÓN [COM]=com(Mr,Ml,n)
% Calculo del centro de masa en la ultima articulacion del robot
%cuando éste posee carga en su efector final. Los paráméros de entrada
%que necesita son la matriz Mr, que es la matriz de 4xn constituida %por
los centros de masa y la masa de cada elemento, y Ml, que es un %vector
fila de orden 4 correspondiente al centro de masa y la masa de %la carga
que se encuentra sujeta al efector final.
if nargin == 2
    n=length(Mr(:,1));
end
if nargin > 3
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 2
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
M=zeros(n,4);
for i=1:n
    for j=1:4
        if i<n
           M(i,j)=Mr(i,j);
        else
           if j<4

M(i,j)=(Mr(i,j)*Mr(i,4)+Ml(1,j)*Ml(1,4))/(Mr(1,4)+Ml(1,4));
           else
               M(i,j)=Mr(i,4)+Ml(1,4);
           end
        end
    end
end
COM=M;
```

APENDICE B.1.3

FUNCIÓN d.m

```
FUNCIÓN [D]=d(U,J,n)
global D
%D   matrices de pseudoinercias.
%
%     D=D(U,J,N) Retorna una matriz de inercias de cada elemento del
%robot. 'U' es un arreglo matricial de variaciones de matrices de paso
%homogéneas respecto a las variables articulares, 'J' es la matriz de
%las pseudoinercias de los elementos del robot y 'N' en el numero de
%grados de libertad o de articulaciones en el robot.
%
%   Copyright (c) 2010 por TMM.
if nargin==2
    n=length(J(1,1,:));
end
if nargin > 3
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 2
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
DD=sym(zeros(n));
for i=1:n
    for j=1:n
        for k=max(i,j):n
            DD(i,j)=DD(i,j)+trace(U(:,:,k,j)*J(:,:,k)*U(:,:,k,i)');
        end
    end
end
D=DD;
```

APENDICE B.1.4

FUNCIÓN genvar.m

```
FUNCIÓN [Q VARQ VVARQ] = genvar(ta)
%GENVAR Genera las variables necesarias para la formulación.
%
%GENVAR(TA) genera dos vectores simbolicos correspondientes a las
%variables articulares y sus derivadas. TA es un vector de caracteres
%que indica el tipo de articulacion. Se debe indicar con 'r' las
%articulaciones rotacionales y con 'p' las articulaciones de %traslación.
if (nargin~=1)
    error('Numero de argumentos invalido.');
end

n=length(ta);
q=sym([]);
varq=sym([]);
vvarq=sym([]);
for i=1:n
    if (ta(i)=='r')
        q(i)=sym(['theta',num2str(i)]);
        varq(i)=sym(['varTheta',num2str(i)]);
        vvarq(i)=sym(['vvarTheta',num2str(i)]);
    elseif (ta(i)=='p')
        q(i)=sym(['d',num2str(i)]);
        varq(i)=sym(['varD',num2str(i)]);
        vvarq(i)=sym(['vvarD',num2str(i)]);
    end
end
Q=q;
VARQ=varq;
VVARQ=vvarq;
```

APENDICE B.1.5

FUNCIÓN h.m

```
FUNCIÓN [HI]=h(HIKM,varq,n)
global HI
%HI   fuerzas de Coriolis y centripetas.
%
%   H=H(HIKM,VARQ,N) Retorna la matriz de las fuerzas centripetas y de
%    coriolis. 'HIKM' es la matriz de terminos HIKM, 'VARQ' es el %vertor
de las variaciones temporales de las variables articulares y %'N' es el
numero de grados de libertad.
%
%   Copyright (c) 2010 por TMM.
if nargin==2
    n=length(varq);
end
if nargin > 3
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 2
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
H=sym(zeros(1,n));
for i=1:n
    for k=1:n
        for m=1:n
            H(i)=H(i)+HIKM(i,k,m)*varq(k)*varq(m);
        end
    end
end
HI=H';
```

APENDICE B.1.6

FUNCIÓN hikm.m

```
FUNCIÓN [HIKM]=hikm(UK,U,J,n)
global hik
%HIKM   terminos hikm.
%
%    H=HIKM(UK,U,J,N) Retorna una matriz de valores HIKM necesarios %para
el calculo de las fuerzas centripetas y de coriolis. 'UK' es un %arreglo
matricial de matrices de las variaciones parciales de U %respecto de cada
variable articular'U' es un arreglo matricial de %variaciones de matrices
de paso homogeneas respecto a las variables %articulares, 'J' es la
matriz de las pseudoinerciasde los elementos %del robot y 'n' en el
numero de grados de libertad o de %articulaciones en el robot.
%
%   Copyright (c) 2010 por TMM.
if nargin==3
    n=length(J(1,1,:));
end
if nargin > 4
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 2
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
H=sym(zeros(n,n,n));
hik=sym(zeros(n,n,n));
for i=1:n
    for k=1:n
        for m=1:n
            for j=max([i k m]):n

H(i,k,m)=H(i,k,m)+trace(UK(:,:,j,k,m)*J(:,:,j)*U(:,:,j,i)');
            end
        end
    end
end
HIKM=H;
hik=H;
```

APENDICE B.1.7

FUNCIÓN ji.m

```
FUNCIÓN [JI]=ji(M,n)

%JI   matrices de pseudoinercias.
%
%   J=JI(M,n) Retorna un arreglo vectorial de matrices de %pseudoinercias
de cada elemento del robot. 'M' es un arreglo en el %que cada iesimo-
componente es un vector con las coordenadas del %centro de masa en
referencia al sistema de coordenadas del elemento.
%
%   Copyright (c) 2010 por TMM.
if nargin == 1
    n=length(M(:,1));
end
if nargin > 2
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 1
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
for i=1:n
    J(1,1,i)=(M(i,1)^2)*M(i,4);
    J(1,2,i)=M(i,1)*M(i,2)*M(i,4);
    J(1,3,i)=M(i,1)*M(i,3)*M(i,4);
    J(1,4,i)=M(i,1)*M(i,4);
    J(2,1,i)=M(i,2)*M(i,1)*M(i,4);
    J(2,2,i)=(M(i,2)^2)*M(i,4);
    J(2,3,i)=M(i,2)*M(i,3)*M(i,4);
    J(2,4,i)=M(i,2)*M(i,4);
    J(3,1,i)=M(i,3)*M(i,1)*M(i,4);
    J(3,2,i)=M(i,3)*M(i,2)*M(i,4);
    J(3,3,i)=(M(i,3)^2)*M(i,4);
    J(3,4,i)=M(i,3)*M(i,4);
    J(4,1,i)=M(i,1)*M(i,4);
    J(4,2,i)=M(i,2)*M(i,4);
    J(4,3,i)=M(i,3)*M(i,4);
    J(4,4,i)=M(i,4);
end
JI=J;
```

APENDICE B.1.8

Script lageu.m

```
% Algoritmo de Lagrange-Euler para la obtencion del modelo dinamico %del
robot
%Parametros previamente establecidos: DH(nx4)'[theta1 d1 a1 alfa1;...]',
q(1xn), varq(1xn), M(nx4)'[xm1 ym1 zm1 m1;...]', g(1x4)'[gx gy gz 0]'
global tl TT

n=length(DH(:,1));
TT=zeros(n,1);
U=uij(DH,q,n);
UK=uijk(DH,q,n);
M=com(Mr,Ml);
J=ji(M,n);
D=d(U,J,n)
HIKM=hikm(UK,U,J,n);
H=h(HIKM,varq,n)
C=c(DH,M,g,U,n)
tl=D*vvarq'+H+C
```

APENDICE B.1.9

FUNCIÓN mph.m

```
FUNCIÓN [MP]= mph(theta,d,a,alfa)
%MPH    matrices de paso homogeneas de las articulaciones del robot.
%
%     A=MPH(THETA, D, A, ALFA) retorna una matriz de paso homogénea
%producto de los valores de Denavit-Hartenberg. THETA es el parametro %de
rotación en torno al eje Zi, D el parametro de desplazamiento a lo %largo
de Zi,
%    A es el parametro de desplazamiento a lo largo de Xj y ALFA es el
%    parametro de desplazamiento en torno al eje Xj. Puede tratar la
%    solucion tanto numericamente como simbolicamente.
%
%    A=MPH(H) retorna una matriz de transformacion del producto
%      de los valores de Denavit-Hartenberg. H es la matriz con los
%parámetros  Denavit-Hartenberg.  Puede  tratar  la  solución  tanto
%numericamente como simbolicamente.
%
%      A=MPH(DH,I,J) retorna una matriz de paso homogenea desde la
%articulación I hasta la articulacion especificada J (AIJ). DH Es una
%matriz que contiene a los elementos de Denavit-Hartenberg donde la
%primera fila corresponde con la primera articulación y así
%sucesivamente. I es la articulación concerniente.
%
%    >> M=AIJ(THETA,D,A,ALFA)
%
%    Copyright (c) 2010 por TMM.

%Etapa de verificacion del numero de argumentos de entrada
if nargin > 4
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 1
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
if nargin == 2
    error('Esta función no admite dos argumentos.');
end

%Contrucción de la matriz de salida para 4 argumentos de entrada.
if nargin == 4
    %Aqui es donde se crea la matriz de paso homogenea.
    A=rotz(theta)*trasl(0,0,d)*trasl(a,0,0)*rotx(alfa);
end

%Contrucción de la matriz de salida para 1 argumento de entrada.
if nargin == 1
    n=length(theta(:,1));
```

```
    %Aqui se comprueba que la matriz con los valores de Denevit-
Hartenberg
    %tenga cuatro columnas y al menos una fila.
    if length(theta(1,:)) ~= 4 || length(theta(:,1)) < 1
        error('Argumento invalido. Debe ser un vector de 4 elementos.');
    end
    %Aqui es donde se crea la matriz de paso homogenea.
    A=eye(4);
    for i=1:n

A=A*rotz(theta(i,1))*trasl(0,0,theta(i,2))*trasl(theta(i,3),0,0)*rotx(the
ta(i,4));
    end
end

%Contrucción de la matriz de salida para 3 argumentos de entrada.
if nargin == 3
    %Aqui se comprueba que la matriz con los valores de Denevit-
Hartenberg
    %tenga cuatro columnas
    if length(theta(1,:)) ~= 4
        error('Argumento invalido. Numero de elementos del vector
insuficientes.');
    end
    %Aqui se comprueba que la matriz con los valores de Denevit-
Hartenberg
    %tenga las articulaciones correspondientes.
    if d > length(theta(:,1)) || a > length(theta(:,1))
        error('Argumento invalido. No existen tantas articulaciones.');
    end
    A=eye(4); %Creación de una matriz identidad.
     for i=min([a d])+1:max([a d])
        %Aqui es donde se va creando la matriz de paso homogenea.

A=A*rotz(theta(i,1))*trasl(0,0,theta(i,2))*trasl(theta(i,3),0,0)*rotx(the
ta(i,4));
     end
     %Crea la matriz de paso en caso de que el orden de transformacion
sea
     %inverso.
     if d > a
        A=inv(A);
     end
end
%Salida de la función.
MP=A;
```

APENDICE B.1.10

Script nweu .m

```
% Algoritmo de Newton-Euler para la obtencion del modelo dinamico del
% robot
%Parametros previamente establecidos: DH(nx4)'[theta1 d1 a1 alfa1;...]',
q(1xn), varq(1xn), M(nx4)'[xm1 ym1 zm1 m1;...]', g(1x4)'[gx gy gz 0]'
global tn w varw varv a
n=length(DH(:,1));
for i=1:n
    RIJ(:,:,1,i)=rot(DH,i-1,i);
    RIJ(:,:,2,i)=RIJ(:,:,1,i)';
end
clear i;
w00=zeros(3,1);    %Base fija o definida por usuario
varw00=zeros(3,1); %Base fija o definida por usuario
v00=zeros(3,1);    %Base fija o definida por usuario
varv00=-g(1:3)';
fn=zeros(3,1);  %Fuerza ejercida sobre el extremo
nn=zeros(3,1);  %Par ejercido sobre elexremo
z0=[0 0 1]';
I=zeros(3,3,n);          %Asumiendo masas concentradas
PP=sym(zeros(n,3));
S=zeros(n,3);
for i=1:n
    PP(i,:)=[DH(i,3);DH(i,2)*sin(DH(i,4));DH(i,2)*cos(DH(i,4))]';
    S(i,:)=M(i,1:3);
end
P=simple(PP(:,1:3));
clear i PP;

w=sym([]);
varw=sym([]);
varv=sym([]);
a=sym([]);
[w varw varv a]=vwag(w00,varw00,varv00,z0,P,S,q,varq,vvarq,RIJ);

tn=sym([]);
ft=sym([]);
nt=sym([]);
for i=n:-1:1
    if i==n
        F=fn;
        N=nn;
        R=eye(3);
        R1=eye(3);
    else
        F=ft(i+1,:)';
        N=nt(i+1,:)';
        R=RIJ(:,:,1,i+1);
        R1=RIJ(:,:,2,i+1);
```

```
    end
    ft(i,:)=(R*F+M(i,4)*a(i,:)')';

nt(i,:)=(R*(N+cross(R1*P(i,:)',F))+cross(P(i,:)'+S(i,:)',M(i,4)*a(i,:)')+
I(:,:,i)*varw(i,:)'+cross(w(i,:)',I(:,:,i)*w(i,:)'))';
    if q(i)==['theta',num2str(i)]
        T = nt(i,:)*RIJ(:,:,2,i)*z0;
        tn(i,1)=T;
        eval(['T' num2str(i) '=T'])
        clear T
    elseif q(i)==['d',num2str(i)]
        F = ft(i,:)*RIJ(:,:,2,i)*z0;
        tn(i,1)=F;
        eval(['F' num2str(i) '=F'])
        clear F
    end
end
tn=tn
clear i N R R1;
```

APENDICE B.1.11

FUNCIÓN rot.m

```
FUNCIÓN [ROT]=rot(DH,i,j)
global ROT
%ROT retorna la matriz de rotacion no homogeneizada entre dos
%articulaciones cualesquiera.
if nargin~=3
    error('Numero de argumentos invalido.');
end
if i<=j
    r=mph(DH,i,j);
    r=r(1:3,1:3);
else
    r=mph(DH,j,i);
    r=r(1:3,1:3);
    r=r';
end
ROT=r;
```

APENDICE B.1.12

FUNCIÓN rotx.m

```
FUNCIÓN [ROTX]=rotx(alfa)
%ROTX     Rotacion en torno al eje X.
%
%   ROTX(ALFA) retorna una matriz de orden cuatro (4) siendo esta una
%    transformación homogenea absoluta producto de una rotación en %torno
al eje X. Donde ALFA es el angulo de rotacion en grados y puede %tratarse
numericamente o simbolicamente.
%
%    >> M=ROTX(ALFA)
%    >> M
%    >> [r][0 0 0]'
%       [0 0 0][1]
%
%   Donde r es la matriz de rotación no homogeneizada de orden tres (3).
%
%   Copyright (c) 2010 por TMM.

%Etapa de verificacion del numero de argumentos de entrada
if nargin > 1
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 1
    error('El numero de argumentos es inferior al requerido por esta
función.');
end

%Etapa de conversión de grados a radianes.
alfa=alfa*pi/180;

%Contrucción de la matriz de salida.
ROTX=[1 0          0           0;
      0 cos(alfa) -sin(alfa)  0;
      0 sin(alfa)  cos(alfa)  0;
      0 0          0          1];
```

APENDICE B.1.13

FUNCIÓN rotz.m

```
FUNCIÓN [ROTZ]=rotz(theta)
%ROTZ     Rotacion en torno al eje Z.
%
%   ROTZ(THETA) retorna una matriz de orden cuatro (4) siendo esta una
%    transformación homogenea absoluta producto de una rotación en %torno
al eje Z. Donde THETA es el angulo de rotacion en grados y %puede
tratarse numericamente o simbolicamente.
%
%    >> M=ROTZ(THETA)
%    >> M
%    >> [r][0 0 0]'
%       [0 0 0][1]
%
%   Donde r es la matriz de rotación no homogeneizada de orden tres (3).
%
%   Copyright (c) 2010 por TMM.

%Etapa de verificacion del numero de argumentos de entrada
if nargin > 1
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 1
    error('El numero de argumentos es inferior al requerido por esta
función.');
end

%Etapa de conversión de grados a radianes.
theta=theta*pi/180;

%Contrucción de la matriz de salida.
ROTZ=[cos(theta) -sin(theta) 0 0;
      sin(theta) cos(theta)  0 0;
      0          0           1 0;
      0          0           0 1];
```

APENDICE B.1.14

FUNCIÓN trasl.m

```
FUNCIÓN [TH]=trasl(x,y,z)
%TRASL     Traslación pura.
%
%    T=TRASL(X,Y,Z) retorna una matriz de orden cuatro (4) siendo esta
%una transformación homogenea absoluta producto de una traslación en %X,
Y y Z. Puede tratar la solucion tanto numericamente como %simbolicamente.
%
%   T=TRASL(T) retorna una matriz de orden cuatro (4) siendo esta una
%   transformación homogenea absoluta producto de una traslación en
%   X, Y y Z. Donde T es el vector desplazamiento en el espacio (x,y,z).
%   Puede tratar la solucion tanto numericamente como simbolicamente.
%
%   >> M=TRASL(X,Y,Z)
%   >> M
%   >> [I][X Y Z]'
%      [0 0 0][1]
%
%   >> M=TRASL(T)
%   >> M
%   >> [I]T'
%      [0 0 0][1]
%   Donde I es la matriz identidad de orden tres (3) y [X Y Z]' o T' %es
el vector de traslación.
%   Copyright (c) 2010 por TMM.
%Etapa de verificacion del numero de argumentos de entrada
if nargin > 3
    error('El numero de argumentos excede el requerido por esta
función.');
end
if nargin < 1
    error('El numero de argumentos es inferior al requerido por esta
función.');
end
if nargin == 2
    error('Esta función no adite dos argumentos.');
end
%Contrucción de la matriz de salida para el caso de uno o tres argumentos
de entrada.
if nargin==1
    T=[1 0 0 x(1);
       0 1 0 x(2);
       0 0 1 x(3);
       0 0 0 1];
else
    T=[1 0 0 x;
       0 1 0 y;
       0 0 1 z;
       0 0 0 1];
end
TH=T;
```

APENDICE B.2- Scripts de las GUI`S para modelación dinámica.

APENDICE B.2.1

Script cgui.m

```
FUNCIÓN varargout = cgui(varargin)
% CGUI M-file for cgui.fig
%      CGUI, by itself, creates a new CGUI or raises the existing
%      singleton*.
%
%      H = CGUI returns the handle to a new CGUI or the handle to
%      the existing singleton*.
%
%      CGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in CGUI.M with the given input arguments.
%
%      CGUI('Property','Value',...) creates a new CGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before cgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to cgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help cgui

% Last Modified by GUIDE v2.5 27-Jun-2011 22:06:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @cgui_OpeningFcn, ...
                   'gui_OutputFcn',  @cgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT
% --- Executes just before cgui is made visible.
FUNCIÓN cgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to cgui (see VARARGIN)

% Choose default command line output for cgui
handles.output = hObject;

%generacion de cuales son las condiciones iniciales
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text2, 'string', ['theta1=']);
            else
                set(handles.text2, 'string', ['d1=']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text3, 'string', ['theta2=']);
            else
                set(handles.text3, 'string', ['d2=']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                set(handles.text4, 'string', ['theta3=']);
            else
                set(handles.text4, 'string', ['d3=']);
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes cgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = cgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
```

```
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ui CI
a=str2double(get(handles.edit1, 'string'));
b=str2double(get(handles.edit2, 'string'));
c=str2double(get(handles.edit3, 'string'));
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            eval(['theta',num2str(i),'=AA']);
        else
            AA=a;
            eval(['d',num2str(i),'=AA']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                BB=b;
                eval(['theta',num2str(i),'=BB']);
            else
                BB=b;
                eval(['d',num2str(i),'=BB']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                CC=c;
                eval(['theta',num2str(i),'=CC']);
        else
                CC=c;
```

```
                eval(['d',num2str(i),'=CC']);
        end
    end

end
ui=evalin('base','uij(DH,q)');
U=ui;
assignin('base','U',ui);
M=evalin('base','com(Mr,Ml)');
assignin('base','M',M)
CI=evalin('base','c(DH,M,g,U)');
cc=eval(CI);
set(handles.text1, 'string', num2str(cc,'%3.2f'));
set(handles.text5, 'string','C=');
```

```
FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
```

```
% --- Executes during object creation, after setting all properties.
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double
% --- Executes during object creation, after setting all properties.
FUNCIÓN edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()
```

```
% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()
```

```
% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.2

Script condinigui.m

```
 FUNCIÓN varargout = condinigui(varargin)
% CONDINIGUI M-file for condinigui.fig
%      CONDINIGUI, by itself, creates a new CONDINIGUI or raises the
existing
%      singleton*.
%
%      H = CONDINIGUI returns the handle to a new CONDINIGUI or the
handle to
%      the existing singleton*.
%
%      CONDINIGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      FUNCIÓN named CALLBACK in CONDINIGUI.M with the given input
arguments.
%
%      CONDINIGUI('Property','Value',...) creates a new CONDINIGUI or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before condinigui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to condinigui_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help condinigui

% Last Modified by GUIDE v2.5 27-Jun-2011 21:40:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @condinigui_OpeningFcn, ...
                   'gui_OutputFcn',  @condinigui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before condinigui is made visible.
FUNCIÓN condinigui_OpeningFcn(hObject, eventdata, handles, varargin)
global celdaCond
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to condinigui (see VARARGIN)

% Choose default command line output for condinigui
handles.output = hObject;

set(handles.uitable1,'Data',celdaCond);
set(handles.uitable1,'ColumnEditable',true(1:5));

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes condinigui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = condinigui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN uitable1_CreateFcn(hObject, eventdata, handles)



% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)

close
modelogui()

% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
```

```
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in siguiente.
FUNCIÓN siguiente_Callback(hObject, eventdata, handles)
% hObject    handle to siguiente (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global tl TT tn band bande NN

cond=get(handles.uitable1,'Data');
n=evalin('base','n');
q=evalin('base','q');
for i=1:n
    qq=cell2mat(cond(i,1));
    vqq=cell2mat(cond(i,2));
    vvqq=cell2mat(cond(i,3));
    if q(i)==['theta',num2str(i)]
        A=qq;
        eval(['theta',num2str(i),'=A']);
        B=vqq;
        eval(['varTheta',num2str(i),'=B']);
        C=vvqq;
        eval(['vvarTheta',num2str(i),'=C']);
    else
        A=qq;
        eval(['d',num2str(i),'=A']);
        B=vqq;
        eval(['varD',num2str(i),'=B']);
        C=vvqq;
        eval(['vvarD',num2str(i),'=C']);
    end
end
if band==1
TT=eval(tl);
end
if bande==1
    NN=eval(tn);
end

close condinigui


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()


% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN asistencia_Callback(hObject, eventdata, handles)
% hObject    handle to asistencia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()
% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
 t_d()
% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()
% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()
% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

 % --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

 % --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()

% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
t_numart()

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.3

Script dgui.m

```
FUNCIÓN varargout = dgui(varargin)
% DGUI M-file for dgui.fig
%      DGUI, by itself, creates a new DGUI or raises the existing
%      singleton*.
%
%      H = DGUI returns the handle to a new DGUI or the handle to
%      the existing singleton*.
%
%      DGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in DGUI.M with the given input arguments.
%
%      DGUI('Property','Value',...) creates a new DGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before dgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to dgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help dgui

% Last Modified by GUIDE v2.5 27-Jun-2011 22:05:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @dgui_OpeningFcn, ...
                   'gui_OutputFcn',  @dgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before dgui is made visible.
FUNCIÓN dgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to dgui (see VARARGIN)

% Choose default command line output for dgui
handles.output = hObject;

%generacion de cuales son las condiciones iniciales
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text2, 'string', 'theta1=');
            else
                set(handles.text2, 'string', 'd1=');
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text3, 'string', 'theta2=');
            else
                set(handles.text3, 'string', 'd2=');
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                set(handles.text4, 'string', 'theta3=');
            else
                set(handles.text4, 'string', 'd3=');
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes dgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = dgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=str2double(get(handles.edit1, 'string'));
b=str2double(get(handles.edit2, 'string'));
c=str2double(get(handles.edit3, 'string'));
q=evalin('base', 'q');
ui=evalin('base','uij(DH,q)');
M=evalin('base','com(Mr,Ml)');
assignin('base','M',M);
JI=evalin('base','ji(M)');
U=ui;
Ji=JI;
assignin('base','U',ui);
assignin('base','J',Ji);
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            eval(['theta',num2str(i),'=AA']);
        else
            AA=a;
            eval(['d',num2str(i),'=AA']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                BB=b;
                eval(['theta',num2str(i),'=BB']);
            else
                BB=b;
```

```
                eval(['d',num2str(i),'=BB']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                CC=c;
                eval(['theta',num2str(i),'=CC']);
        else
                CC=c;
                eval(['d',num2str(i),'=CC']);
        end
    end

end
D=evalin('base','d(U,J)');
dd=eval(D);
set(handles.text1, 'string', num2str(dd,'%3.1f    '));
set(handles.text5, 'string',['D=']);
```

```
FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
FUNCIÓN Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.4

Script DiBotMat.m

```
FUNCIÓN varargout = DiBotMat(varargin)
% AYUDA M-file for Ayuda.fig
%      AYUDA, by itself, creates a new AYUDA or raises the existing
%      singleton*.
%
%      H = AYUDA returns the handle to a new AYUDA or the handle to
%      the existing singleton*.
%
%      AYUDA('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in AYUDA.M with the given input arguments.
%
%      AYUDA('Property','Value',...) creates a new AYUDA or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before DiBotMat_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to DiBotMat_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Ayuda

% Last Modified by GUIDE v2.5 27-Jun-2011 21:21:11

% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DiBotMat_OpeningFcn, ...
                   'gui_OutputFcn',  @DiBotMat_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Ayuda is made visible.
FUNCIÓN DiBotMat_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Ayuda (see VARARGIN)

%Imagen de fondo
im=imread('principal.jpg');
image(im)
axis off

%Coloca una imagen en cada botón
[a,map]=imread('adelante.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/65);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.continuar,'CData',g);
% Choose default command line output for Ayuda
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Ayuda wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = DiBotMat_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in continuar.
FUNCIÓN continuar_Callback(hObject, eventdata, handles)
% hObject    handle to continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
robotgui()


% --- Executes during object creation, after setting all properties.
FUNCIÓN axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_DH()

% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()

% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()

% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()

% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()

% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()

% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()

% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()

% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()
 % --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
t_numart()
% --- Executes when user attempts to close figure1.
FUNCIÓN figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(hObject);

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.5

Script funciongui .m

```
FUNCIÓN varargout = funciongui(varargin)
% FUNCIONGUI M-file for funciongui.fig
%      FUNCIONGUI, by itself, creates a new FUNCIONGUI or raises the
existing
%      singleton*.
%
%      H = FUNCIONGUI returns the handle to a new FUNCIONGUI or the
handle to
%      the existing singleton*.
%
%      FUNCIONGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      FUNCIÓN named CALLBACK in FUNCIONGUI.M with the given input
arguments.
%
%      FUNCIONGUI('Property','Value',...) creates a new FUNCIONGUI or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before funciongui_OpeningFcn gets called.  An
```

```
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to funciongui_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help funciongui

% Last Modified by GUIDE v2.5 27-Jun-2011 21:47:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @funciongui_OpeningFcn, ...
                   'gui_OutputFcn',  @funciongui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before funciongui is made visible.
FUNCIÓN funciongui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to funciongui (see VARARGIN)

% Choose default command line output for funciongui
handles.output = hObject;

%Imagen de fondo
im=imread('demas.jpg');
image(im)
axis off

%Coloca una imagen en cada botón
[a,map]=imread('atras.jpg');
```

```
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/65);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.atras,'CData',g);

%Coloca una imagen en cada botón
[a,map]=imread('ayuda.jpg');
[r,c,d]=size(a);
x=ceil(r/31);
y=ceil(c/31);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.ayud5,'CData',g);
set(handles.ayud6,'CData',g);
set(handles.ayud7,'CData',g);
set(handles.ayud8,'CData',g);
set(handles.ayud9,'CData',g);
set(handles.ayud10,'CData',g);
set(handles.ayud11,'CData',g);
set(handles.ayud12,'CData',g);


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes funciongui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = funciongui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in atras.
FUNCIÓN atras_Callback(hObject, eventdata, handles)
% hObject    handle to atras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
modelogui()



% --- Executes during object creation, after setting all properties.
```

```
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)

tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in mphh.
FUNCIÓN mphh_Callback(hObject, eventdata, handles)
% hObject    handle to mphh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mphgui()

% --- Executes on button press in uijj.
FUNCIÓN uijj_Callback(hObject, eventdata, handles)
% hObject    handle to uijj (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
q=evalin('base','q');
i=length(q);

    if i==1
        uij1();
    end
    if i==2
        uij2();
    end
    if i==3
        uij3();
    end




% --- Executes on button press in hii.
FUNCIÓN hii_Callback(hObject, eventdata, handles)
% hObject    handle to hii (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
hgui()


% --- Executes on button press in dii.
FUNCIÓN dii_Callback(hObject, eventdata, handles)
% hObject    handle to dii (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dgui()
```

```
% --- Executes on button press in jii.
FUNCIÓN jii_Callback(hObject, eventdata, handles)
% hObject    handle to jii (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
q=evalin('base', 'q');
i=length(q);

    if i==1
        ji1();
    end
    if i==2
        ji2();
    end
    if i==3
        ji3();
    end



% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()


% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()
```

```
% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()
```

```
% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


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

```
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
t_numart()


% --- Executes on button press in vwagg.
FUNCIÓN vwagg_Callback(hObject, eventdata, handles)
% hObject    handle to vwagg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
vwagui()


% --- Executes on button press in rott.
FUNCIÓN rott_Callback(hObject, eventdata, handles)
% hObject    handle to rott (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
rotgui()


% --- Executes on button press in cii.
FUNCIÓN cii_Callback(hObject, eventdata, handles)
% hObject    handle to cii (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cgui()


% --- Executes during object creation, after setting all properties.
FUNCIÓN cii_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cii (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes on button press in ayud12.
FUNCIÓN ayud12_Callback(hObject, eventdata, handles)
% hObject    handle to ayud12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag
```

```
% --- Executes on button press in ayud11.
FUNCIÓN ayud11_Callback(hObject, eventdata, handles)
% hObject    handle to ayud11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot


% --- Executes on button press in ayud10.
FUNCIÓN ayud10_Callback(hObject, eventdata, handles)
% hObject    handle to ayud10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c


% --- Executes on button press in ayud9.
FUNCIÓN ayud9_Callback(hObject, eventdata, handles)
% hObject    handle to ayud9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h


% --- Executes on button press in ayud8.
FUNCIÓN ayud8_Callback(hObject, eventdata, handles)
% hObject    handle to ayud8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_d


% --- Executes on button press in ayud7.
FUNCIÓN ayud7_Callback(hObject, eventdata, handles)
% hObject    handle to ayud7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --- Executes on button press in ayud6.
FUNCIÓN ayud6_Callback(hObject, eventdata, handles)
% hObject    handle to ayud6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --- Executes on button press in ayud5.
FUNCIÓN ayud5_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to ayud5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.6

Script hgui.m

```
FUNCIÓN varargout = hgui(varargin)
% HGUI M-file for hgui.fig
%      HGUI, by itself, creates a new HGUI or raises the existing
%      singleton*.
%
%      H = HGUI returns the handle to a new HGUI or the handle to
%      the existing singleton*.
%
%      HGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in HGUI.M with the given input arguments.
%
%      HGUI('Property','Value',...) creates a new HGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before hgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to hgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help hgui

% Last Modified by GUIDE v2.5 27-Jun-2011 22:10:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @hgui_OpeningFcn, ...
                   'gui_OutputFcn',  @hgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before hgui is made visible.
FUNCIÓN hgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to hgui (see VARARGIN)

% Choose default command line output for hgui
handles.output = hObject;

%generacion de cuales son las condiciones iniciales
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text2, 'string', ['theta1=']);
                set(handles.text5, 'string', ['varTheta1=']);
            else
                set(handles.text2, 'string', ['d1=']);
                set(handles.text5, 'string', ['varD1=']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text3, 'string', ['theta2=']);
                set(handles.text6, 'string', ['varTheta2=']);
            else
                set(handles.text3, 'string', ['d2=']);
```

```
                set(handles.text6, 'string', ['varD2=']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                set(handles.text4, 'string', ['theta3=']);
                set(handles.text7, 'string', ['varTheta3=']);
            else
                set(handles.text4, 'string', ['d3=']);
                set(handles.text7, 'string', ['varD3=']);
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes hgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = hgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ui uik hik JI HI
```

```
a=str2double(get(handles.edit1, 'string'));
b=str2double(get(handles.edit2, 'string'));
c=str2double(get(handles.edit3, 'string'));
d=str2double(get(handles.edit4, 'string'));
e=str2double(get(handles.edit5, 'string'));
f=str2double(get(handles.edit6, 'string'));
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            DD=d;
            eval(['theta',num2str(i),'=AA']);
            eval(['varTheta',num2str(i),'=DD']);
        else
            AA=a;
            DD=d;
            eval(['d',num2str(i),'=AA']);
            eval(['varD',num2str(i),'=DD']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                BB=b;
                EE=e;
                eval(['theta',num2str(i),'=BB']);
                eval(['varTheta',num2str(i),'=EE']);
            else
                BB=b;
                EE=e;
                eval(['d',num2str(i),'=BB']);
                eval(['varD',num2str(i),'=EE']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                CC=c;
                FF=f;
                eval(['theta',num2str(i),'=CC']);
                eval(['varTheta',num2str(i),'=FF']);
        else
                CC=c;
                FF=f;
                eval(['d',num2str(i),'=CC']);
                eval(['varD',num2str(i),'=FF']);
        end
    end

end
ui=evalin('base','uij(DH,q)');
JI=evalin('base','ji(M)');
uik=evalin('base','uijk(DH,q)');
U=ui;
Ji=JI;
```

```
uikk=uik;
assignin('base',['U'],ui);
assignin('base',['J'],Ji);
assignin('base',['UK'],uikk);
hik=evalin('base','hikm(UK,U,J)');
hikk=hik;
assignin('base',['HIKM'],hikk);
HI=evalin('base','h(HIKM,varq)');
hi=eval(HI);
set(handles.text1, 'string', num2str(hi,'%3.1f    '));
set(handles.text8, 'string',['H=']);



FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


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

```
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.7

Script ji1.m

```
FUNCIÓN varargout = ji1(varargin)
% JI1 M-file for ji1.fig
%      JI1, by itself, creates a new JI1 or raises the existing
%      singleton*.
%
%      H = JI1 returns the handle to a new JI1 or the handle to
%      the existing singleton*.
%
%      JI1('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in JI1.M with the given input arguments.
%
%      JI1('Property','Value',...) creates a new JI1 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ji1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ji1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to help ji1

% Last Modified by GUIDE v2.5 27-Jun-2011 21:59:23

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ji1_OpeningFcn, ...
                   'gui_OutputFcn',  @ji1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ji1 is made visible.
FUNCIÓN ji1_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ji1 (see VARARGIN)

% Choose default command line output for ji1
handles.output = hObject;
set(handles.text1, 'string','J1=');
M=evalin('base','com(Mr,Ml)');
assignin('base','M',M);
Ji1=evalin('base','ji(M)');

set(handles.text2, 'string', num2str(Ji1,'%3.2f   '));

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ji1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = ji1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


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

```
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()
```

```
% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.8

Script ji2.m

```
FUNCIÓN varargout = ji2(varargin)
% JI2 M-file for ji2.fig
%      JI2, by itself, creates a new JI2 or raises the existing
%      singleton*.
%
%      H = JI2 returns the handle to a new JI2 or the handle to
%      the existing singleton*.
%
%      JI2('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in JI2.M with the given input arguments.
%
%      JI2('Property','Value',...) creates a new JI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ji2_OpeningFcn gets called.  An
```

```
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ji2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ji2

% Last Modified by GUIDE v2.5 27-Jun-2011 22:01:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ji2_OpeningFcn, ...
                   'gui_OutputFcn',  @ji2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ji2 is made visible.
FUNCIÓN ji2_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ji2 (see VARARGIN)

% Choose default command line output for ji2
handles.output = hObject;

set(handles.text1, 'string','J1=');
set(handles.text3, 'string','J2=');
M=evalin('base','com(Mr,Ml)');
assignin('base','M',M);
Ji2=evalin('base','ji(M)');
for i=1:2
    if i==1
        Jii=Ji2(:,:,i);
        set(handles.text2, 'string', num2str(Jii,'%3.2f  '));
```

```
    end
    if i==2
        Jii=Ji2(:,:,i);
        set(handles.text4, 'string', num2str(Jii,'%3.2f  '));
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ji2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = ji2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --------------------------------------------------------------------
FUNCIÓN Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()
```

```
% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()

% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()

% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()

% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()

% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()

% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()

% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.9
Script ji3.m

```
FUNCIÓN varargout = ji3(varargin)
% JI3 M-file for ji3.fig
%      JI3, by itself, creates a new JI3 or raises the existing
%      singleton*.
%
%      H = JI3 returns the handle to a new JI3 or the handle to
%      the existing singleton*.
%
```

```
%      JI3('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in JI3.M with the given input arguments.
%
%      JI3('Property','Value',...) creates a new JI3 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ji3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ji3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ji3

% Last Modified by GUIDE v2.5 27-Jun-2011 22:03:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ji3_OpeningFcn, ...
                   'gui_OutputFcn',  @ji3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before ji3 is made visible.
FUNCIÓN ji3_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ji3 (see VARARGIN)

% Choose default command line output for ji3
handles.output = hObject;

%generacion de la matriz ji3
set(handles.text1, 'string','J1=');
set(handles.text3, 'string','J2=');
set(handles.text5, 'string','J3=');
```

```
M=evalin('base','com(Mr,Ml)');
assignin('base','M',M);
Ji=evalin('base','ji(M)');
i=3;
for i=1:3
if i==1
    Jii=Ji(:,:,i);
   set(handles.text2, 'string', num2str(Jii,'%3.2f  '));
end
if i==2
    Jii=Ji(:,:,i);
    set(handles.text4, 'string', num2str(Jii,'%3.2f  '));
end
if i==3
    Jii=Ji(:,:,i);
    set(handles.text6, 'string', num2str(Jii,'%3.2f  '));
end

end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ji3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = ji3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);

% --------------------------------------------------------------------
FUNCIÓN Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()

% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()

% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()

% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()

% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()

% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()

 % --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()

% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()

% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()
```

```
% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()

 % --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

 % --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.10

Script modelogui.m

```
FUNCIÓN varargout = modelogui(varargin)
% MODELOGUI M-file for modelogui.fig
%      MODELOGUI, by itself, creates a new MODELOGUI or raises the
existing
%      singleton*.
%
%      H = MODELOGUI returns the handle to a new MODELOGUI or the handle
to
%      the existing singleton*.
%
%      MODELOGUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      FUNCIÓN named CALLBACK in MODELOGUI.M with the given input
arguments.
%
%      MODELOGUI('Property','Value',...) creates a new MODELOGUI or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before modelogui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to modelogui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help modelogui

% Last Modified by GUIDE v2.5 27-Jun-2011 21:30:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @modelogui_OpeningFcn, ...
                   'gui_OutputFcn',  @modelogui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before modelogui is made visible.
FUNCIÓN modelogui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to modelogui (see VARARGIN)

% Choose default command line output for modelogui
handles.output = hObject;

im=imread('demas.jpg');
image(im)
axis off

%Coloca una imagen en cada botón
[a,map]=imread('atras.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/65);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.atras,'CData',g);

%Coloca una imagen en cada botón
[a,map]=imread('adelante.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/65);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.sigue,'CData',g);

%Coloca una imagen en cada botón
[a,map]=imread('ayuda.jpg');
[r,c,d]=size(a);
x=ceil(r/31);
y=ceil(c/31);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.ayud4,'CData',g);


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes modelogui wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = modelogui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in le.
FUNCIÓN le_Callback(hObject, eventdata, handles)
global band

band=1;
tic;
    evalin('base','lageu');
    t=toc;
    set(handles.text1,'string',t);


% --- Executes on button press in ne.
FUNCIÓN ne_Callback(hObject, eventdata, handles)
global bande

bande=1;
 tic;
    evalin('base','nweu');
    t=toc;
    set(handles.text2,'string',t);


% --- Executes on button press in atras.
FUNCIÓN atras_Callback(hObject, eventdata, handles)

close
robotgui()




% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
global band bande
band=0;
bande=0;
```

```
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in sigue.
FUNCIÓN sigue_Callback(hObject, eventdata, handles)
% hObject    handle to sigue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global band bande

if band==0
    if bande==0
        errordlg('Debe presionar uno o los dos algoritmos','Error en la
selección de tarea');
    else
        close
        resultado()
    end
else
    close
    resultado()
end


% --- Executes during object creation, after setting all properties.
FUNCIÓN text2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% --- Executes on button press in pushbutton6.
FUNCIÓN pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
funciongui()


% --- Executes on button press in ne.
FUNCIÓN pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to ne (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()


% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
%handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()

% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()

% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()

% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()

% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
t_c()

% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()

% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
t_numart()

% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in ayud4.
FUNCIÓN ayud4_Callback(hObject, eventdata, handles)
% hObject    handle to ayud4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

 % --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.11

Script mphgui .m

```
FUNCIÓN varargout = mphgui(varargin)
% MPHGUI M-file for mphgui.fig
%      MPHGUI, by itself, creates a new MPHGUI or raises the existing
%      singleton*.
%
%      H = MPHGUI returns the handle to a new MPHGUI or the handle to
%      the existing singleton*.
%
%      MPHGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in MPHGUI.M with the given input arguments.
%
%      MPHGUI('Property','Value',...) creates a new MPHGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before mphgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to mphgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mphgui

% Last Modified by GUIDE v2.5 27-Jun-2011 21:50:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mphgui_OpeningFcn, ...
                   'gui_OutputFcn',  @mphgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before mphgui is made visible.
FUNCIÓN mphgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mphgui (see VARARGIN)

% Choose default command line output for mphgui
handles.output = hObject;

%generación de cuáles son las condiciones iniciales
set(handles.text1, 'string', ['A']);
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text4, 'string', ['theta1=']);
            else
                set(handles.text4, 'string', ['d1=']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text5, 'string', ['theta2=']);
            else
                set(handles.text5, 'string', ['d2=']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                set(handles.text6, 'string', ['theta3=']);
            else
                set(handles.text6, 'string', ['d3=']);
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mphgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = mphgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);

% --- Executes on button press in pushbutton1.
FUNCIÓN pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
global MP

x=str2double(get(handles.edit2, 'string'));
y=str2double(get(handles.edit1, 'string'));
assignin('base',['x'],x);
assignin('base',['y'],y);
a=str2double(get(handles.edit3, 'string'));
b=str2double(get(handles.edit4, 'string'));
c=str2double(get(handles.edit5, 'string'));
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            eval(['theta',num2str(i),'=AA']);
        else
            AA=a;
            eval(['d',num2str(i),'=AA']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                BB=b;
                eval(['theta',num2str(i),'=BB']);
            else
                BB=b;
                eval(['d',num2str(i),'=BB']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                CC=c;
                eval(['theta',num2str(i),'=CC']);
        else
                CC=c;
                eval(['d',num2str(i),'=CC']);
        end
    end

end
MP=evalin('base','mph(DH,x,y)');
mp=eval(MP);
set(handles.text3, 'string', num2str(mp,'%3.1f    '));

FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
```

```
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

FUNCIÓN edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double
% --- Executes during object creation, after setting all properties.

FUNCIÓN edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

FUNCIÓN edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double

%--- Executes during object creation, after setting all properties.
FUNCIÓN edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double

% --- Executes during object creation, after setting all properties.
FUNCIÓN edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()

% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()

% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()

% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_com()

FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()

% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()

% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()

% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_uij()

% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()

% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()

% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()

% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APENDICE B.2.12

Script resultado.m

```
FUNCIÓN varargout = resultado(varargin)
% RESULTADO M-file for resultado.fig
%      RESULTADO, by itself, creates a new RESULTADO or raises the
existing
%      singleton*.
%
%      H = RESULTADO returns the handle to a new RESULTADO or the handle
to
%      the existing singleton*.
%
%      RESULTADO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      FUNCIÓN named CALLBACK in RESULTADO.M with the given input
arguments.
%
%      RESULTADO('Property','Value',...) creates a new RESULTADO or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before resultado_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to resultado_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help resultado

% Last Modified by GUIDE v2.5 01-Oct-2011 10:48:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @resultado_OpeningFcn, ...
                   'gui_OutputFcn',  @resultado_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before resultado is made visible.
FUNCIÓN resultado_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to resultado (see VARARGIN)

% Choose default command line output for resultado
handles.output = hObject;

im=imread('demas.jpg');
image(im)
axis off

%Coloca una imagen en cada botón
[a,map]=imread('atras.jpg');
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/65);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.atras,'CData',g);


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes resultado wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = resultado_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in pushbutton1.
FUNCIÓN pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
condinigui

uiwait


global TT NN band bande
jt=TT;
jn=NN;
n=evalin('base','n');
q=evalin('base', 'q');

if band==1
    set(handles.text1, 'string', num2str(jt));
    set(handles.text3, 'string',['Torques y/o Fuerzas L-E']);
    for i=1:n
        if i==1
            if q(i)==['theta',num2str(i)]
                set(handles.text2, 'string', ['T1=']);
            else
                set(handles.text2, 'string', ['F1=']);
            end
        end
        if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text7, 'string', ['T2=']);
            else
```

```
                set(handles.text7, 'string', ['F2=']);
            end
        end
        if i==3
            if q(i)==['theta',num2str(i)]
                set(handles.text8, 'string', ['T3=']);
            else
                set(handles.text8, 'string', ['F3=']);
            end
        end
    end
end

if bande==1
    set(handles.text5, 'string', num2str(jn));
    set(handles.text4, 'string',['Torques y/o Fuerzas N-E']);
    for i=1:n
        if i==1
            if q(i)==['theta',num2str(i)]
                set(handles.text6, 'string', ['T1=']);
            else
                set(handles.text6, 'string', ['F1=']);
            end
        end
        if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text9, 'string', ['T2=']);
            else
                set(handles.text9, 'string', ['F2=']);
            end
        end
        if i==3
            if q(i)==['theta',num2str(i)]
                set(handles.text10, 'string', ['T3=']);
            else
                set(handles.text10, 'string', ['F3=']);
            end
        end
    end
end

% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()
```

```
% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()


% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()



% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()
```

```
% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
t_numart()


% --- Executes on button press in atras.
FUNCIÓN atras_Callback(hObject, eventdata, handles)
% hObject    handle to atras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
modelogui()


% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
return
else
winopen(strcat(Path,MANUAL));
end
```

APENDICE B.2.13

Script robotgui.m

```
FUNCIÓN varargout = robotgui(varargin)
% ROBOTGUI M-file for robotgui.fig
%      ROBOTGUI, by itself, creates a new ROBOTGUI or raises the existing
%      singleton*.
%
%      H = ROBOTGUI returns the handle to a new ROBOTGUI or the handle to
%      the existing singleton*.
%
%      ROBOTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in ROBOTGUI.M with the given input
arguments.
%
%      ROBOTGUI('Property','Value',...) creates a new ROBOTGUI or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before robotgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to robotgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```
% Edit the above text to modify the response to a_yuda robotgui

% Last Modified by GUIDE v2.5 26-Sep-2011 12:39:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @robotgui_OpeningFcn, ...
                   'gui_OutputFcn',  @robotgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before robotgui is made visible.
FUNCIÓN robotgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to robotgui (see VARARGIN)

% Choose default command line output for robotgui
handles.output = hObject;

%Imagen de fondo
%Imagen de fondo
im=imread('demas.jpg');
image(im)
axis off

%Coloca una imagen en cada botón
[a,map]=imread('atras.jpg');
[r,c,d]=size(a);
x=ceil(r/39);
y=ceil(c/70);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Volver,'CData',g);

%Coloca una imagen en cada botón
[a,map]=imread('adelante.jpg');
```

```
[r,c,d]=size(a);
x=ceil(r/40);
y=ceil(c/70);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.siguiente,'CData',g);

%Coloca una imagen en cada botón
[a,map]=imread('ayuda.jpg');
[r,c,d]=size(a);
x=ceil(r/31);
y=ceil(c/31);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.ayud,'CData',g);
set(handles.ayud1,'CData',g);
set(handles.ayud2,'CData',g);
set(handles.ayud3,'CData',g);


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes robotgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = robotgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in n.
FUNCIÓN n_Callback(hObject, eventdata, handles)
global celdaCond

nn=get(handles.n,'value');
nn=nn-1;
celdaDH=cell(nn,5);
celdaM=cell(nn,4);
celdaDH(:,:)={0};
celdaM(:,:)={0};
celdaCond=cell(nn,3);
celdaCond(:,:)={0};
for i=1:nn
    celdaDH(i,1)={['THETA' num2str(i)]};
    celdaDH(i,5)={''};
```

```
end
set(handles.DH,'Data',celdaDH);

set(handles.M,'Data',celdaM);
set(handles.M,'ColumnEditable',true(1,5));

% Hints: contents = get(hObject,'String') returns n contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from n


% --- Executes during object creation, after setting all properties.
FUNCIÓN n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when entered data in editable cell(s) in DH.
FUNCIÓN DH_CellEditCallback(hObject, eventdata, handles)
d=handles.celdaActual;
data=get(handles.DH,'Data');
if d(2)==5
    if cell2mat(data(d(1),5))=='D'
        data(d(1),2)={['D' num2str(d(1))]};
        set(handles.DH,'ColumnEditable',false(1,1));
        data(d(1),1)={0};
    else
        data(d(1),1)={['THETA' num2str(d(1))]};
        data(d(1),2)={0};
    end
    set(handles.DH,'Data',data);
end


% --- Executes during object creation, after setting all properties.
FUNCIÓN ventana_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ventana (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --- Executes when selected cell(s) is changed in DH.
FUNCIÓN DH_CellSelectionCallback(hObject, eventdata, handles)

handles.celdaActual=eventdata.Indices;
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
FUNCIÓN DH_CreateFcn(hObject, eventdata, handles)
%Eliminar esta funcion


% --- Executes on selection change in popupmenu2.
FUNCIÓN popupmenu2_Callback(hObject, eventdata, handles)
%Eliminar esta funcion

% --- Executes during object creation, after setting all properties.
FUNCIÓN popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when entered data in editable cell(s) in uitable3.
FUNCIÓN uitable3_CellEditCallback(hObject, eventdata, handles)
%Eliminar esta funcion


FUNCIÓN gx_Callback(hObject, eventdata, handles)
%Eliminar esta funcion


% --- Executes during object creation, after setting all properties.
FUNCIÓN gx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN gy_Callback(hObject, eventdata, handles)
%Eliminar esta funcion

% --- Executes during object creation, after setting all properties.
FUNCIÓN gy_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gy (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN gz_Callback(hObject, eventdata, handles)
%Eliminar esta funcion


% --- Executes during object creation, after setting all properties.
FUNCIÓN gz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)

evalin('base','clear');
evalin('base','clc');
tam=get(0,'ScreenSize');
```

```
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);

% --- Executes on button press in pushbutton3.
FUNCIÓN pushbutton3_Callback(hObject, eventdata, handles)
%Eliminar esta funcion

% --- Executes during object creation, after setting all properties.
FUNCIÓN aceptar1_CreateFcn(hObject, eventdata, handles)
%Eliminar esta funcion


% --- Executes on button press in Volver.
FUNCIÓN Volver_Callback(hObject, eventdata, handles)
% hObject    handle to Volver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
DiBotMat()


% --- Executes on button press in siguiente.
FUNCIÓN siguiente_Callback(hObject, eventdata, handles)
n=get(handles.n,'value');
n=n-1;
if n==0
    errordlg('Asegurese de ingresar todos los datos necesarios.','error
de ingreso de datos');
else
    dataDH=get(handles.DH,'Data');
    for i=1:n
        if cell2mat(dataDH(i,5))=='D'
            q(i)=sym(['d',num2str(i)]);
            varq(i)=sym(['varD',num2str(i)]);
            vvarq(i)=sym(['vvarD',num2str(i)]);
        else
            q(i)=sym(['theta',num2str(i)]);
            varq(i)=sym(['varTheta',num2str(i)]);
            vvarq(i)=sym(['vvarTheta',num2str(i)]);
        end
    end
    dataM=get(handles.M,'data');
    M=sym([]);
    DH=sym([]);
    for i=1:n
        for j=1:4
            if ~isnumeric(cell2mat(dataM(i,j)))
                M(i,j)=sym(cell2mat(dataM(i,j)));
            else
                M(i,j)=cell2mat(dataM(i,j));
```

```
            end
            if ~isnumeric(cell2mat(dataDH(i,j)))
                DH(i,j)=q(i);
            else
                DH(i,j)=cell2mat(dataDH(i,j));
            end
        end
    end
    gx=get(handles.gx,'String');
    gy=get(handles.gy,'String');
    gz=get(handles.gz,'String');
    gx=str2double(gx);
    gy=str2double(gy);
    gz=str2double(gz);
    g=[gx gy gz 0];
    loadx=get(handles.loadx,'String');
    loady=get(handles.loady,'String');
    loadz=get(handles.loadz,'String');
    ml=get(handles.ml,'String');
    loadx=str2double(loadx);
    loady=str2double(loady);
    loadz=str2double(loadz);
    ml=str2double(ml);
    Ml=[loadx loady loadz ml];
    assignin('base','DH',DH);
    assignin('base','q',q);
    assignin('base','varq',varq);
    assignin('base','vvarq',vvarq);
    assignin('base','Mr',M);
    assignin('base','g',g);
    assignin('base','Ml',Ml);
    close
    modelogui()
end


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()


% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()
```

```
% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()



% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA
t_numart()

% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when user attempts to close figure1.
FUNCIÓN figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(hObject);


% --- Executes on button press in ayud2.
FUNCIÓN ayud2_Callback(hObject, eventdata, handles)
% hObject    handle to ayud2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --- Executes on button press in ayud.
FUNCIÓN ayud_Callback(hObject, eventdata, handles)
% hObject    handle to ayud (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --- Executes on button press in ayud1.
FUNCIÓN ayud1_Callback(hObject, eventdata, handles)
% hObject    handle to ayud1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()


% --- Executes on button press in ayud3.
FUNCIÓN ayud3_Callback(hObject, eventdata, handles)
% hObject    handle to ayud3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton10.
FUNCIÓN pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



FUNCIÓN loadx_Callback(hObject, eventdata, handles)
% hObject    handle to loadx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of loadx as text
%        str2double(get(hObject,'String')) returns contents of loadx as a
double
```

```
% --- Executes during object creation, after setting all properties.
FUNCIÓN loadx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN loady_Callback(hObject, eventdata, handles)
% hObject    handle to loady (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of loady as text
%        str2double(get(hObject,'String')) returns contents of loady as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN loady_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loady (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

FUNCIÓN loadz_Callback(hObject, eventdata, handles)
% hObject    handle to loadz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of loadz as text
%        str2double(get(hObject,'String')) returns contents of loadz as a
double

% --- Executes during object creation, after setting all properties.
FUNCIÓN loadz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to loadz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

FUNCIÓN ml_Callback(hObject, eventdata, handles)
% hObject    handle to ml (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of ml as text
%        str2double(get(hObject,'String')) returns contents of ml as a
double

% --- Executes during object creation, after setting all properties.
FUNCIÓN ml_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ml (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
return
else
winopen(strcat(Path,MANUAL));
end
```

APENDICE B.2.14

Script rotgui.m

```
FUNCIÓN varargout = rotgui(varargin)
% ROTGUI M-file for rotgui.fig
%      ROTGUI, by itself, creates a new ROTGUI or raises the existing
%      singleton*.
%
%      H = ROTGUI returns the handle to a new ROTGUI or the handle to
```

```
%      the existing singleton*.
%
%      ROTGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in ROTGUI.M with the given input arguments.
%
%      ROTGUI('Property','Value',...) creates a new ROTGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before rotgui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to rotgui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help rotgui

% Last Modified by GUIDE v2.5 01-Oct-2011 11:34:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @rotgui_OpeningFcn, ...
                   'gui_OutputFcn',  @rotgui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before rotgui is made visible.
FUNCIÓN rotgui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to rotgui (see VARARGIN)

% Choose default command line output for rotgui
handles.output = hObject;
```

```
%generaciosn de cuales son las condiciones iniciales
set(handles.text1, 'string', ['R']);
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text3, 'string', ['theta1=']);
            else
                set(handles.text3, 'string', ['d1=']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text4, 'string', ['theta2=']);
            else
                set(handles.text4, 'string', ['d2=']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                set(handles.text5, 'string', ['theta3=']);
            else
                set(handles.text5, 'string', ['d3=']);
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes rotgui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = rotgui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
```

```
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global MP

x=str2double(get(handles.edit2, 'string'));
y=str2double(get(handles.edit1, 'string'));
assignin('base',['x'],x);
assignin('base',['y'],y);
a=str2double(get(handles.edit3, 'string'));
b=str2double(get(handles.edit4, 'string'));
c=str2double(get(handles.edit5, 'string'));
q=evalin('base', 'q');
n=length(q);
for i=1:n
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            eval(['theta',num2str(i),'=AA']);
```

```
        else
            AA=a;
            eval(['d',num2str(i),'=AA']);
        end
    end
    if i==2
            if q(i)==['theta',num2str(i)]
                BB=b;
                eval(['theta',num2str(i),'=BB']);
            else
                BB=b;
                eval(['d',num2str(i),'=BB']);
            end
    end
    if i==3
        if q(i)==['theta',num2str(i)]
                CC=c;
                eval(['theta',num2str(i),'=CC']);
        else
                CC=c;
                eval(['d',num2str(i),'=CC']);
        end
    end

end
ROT=evalin('base','rot(DH,x,y)');
rot=eval(ROT);
set(handles.text2, 'string', num2str(rot,'%3.1f    '));



FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()
```

```
% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()
```

```
% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()

% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
return
else
winopen(strcat(Path,MANUAL));
end
```

APENDICE B.2.15

Script uij1.m

```
FUNCIÓN varargout = uij1(varargin)
% UIJ1 M-file for uij1.fig
%      UIJ1, by itself, creates a new UIJ1 or raises the existing
%      singleton*.
%
%      H = UIJ1 returns the handle to a new UIJ1 or the handle to
%      the existing singleton*.
%
%      UIJ1('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in UIJ1.M with the given input arguments.
%
%      UIJ1('Property','Value',...) creates a new UIJ1 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before uij1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to uij1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help uij1

% Last Modified by GUIDE v2.5 03-Oct-2011 20:16:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @uij1_OpeningFcn, ...
                   'gui_OutputFcn',  @uij1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before uij1 is made visible.
```

```
FUNCIÓN uij1_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to uij1 (see VARARGIN)

% Choose default command line output for uij1
handles.output = hObject;

%generacion de las condiciones iniciales
q=evalin('base', 'q');
i=1;
 if q(i)==['theta',num2str(i)]
                set(handles.text1, 'string', ['theta1=']);
            else
                set(handles.text1, 'string', ['d1=']);
            end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes uij1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = uij1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);
```

```
FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ui
uii=ui;
q=evalin('base','q');
uii=evalin('base','uij(DH,q)');
a=str2double(get(handles.edit1, 'string'));
i=1;
if q(i)==['theta',num2str(i)]
    AA=a;
    eval(['theta',num2str(i),'=AA']);
else
    AA=a;
    eval(['d',num2str(i),'=AA']);
end
uii=eval(uii);

set(handles.text3, 'string', num2str(uii,'%3.1f    '));
set(handles.text2, 'string', ['U11=']);

% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()

% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()

% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()

% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()

% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_d()
% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()

% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()

% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()

% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()

% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------

FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
```

```
return
else
winopen(strcat(Path,MANUAL));
end
```

APENDICE B.2.16

Script uij2.m

```
FUNCIÓN varargout = uij2(varargin)
% UIJ2 M-file for uij2.fig
%      UIJ2, by itself, creates a new UIJ2 or raises the existing
%      singleton*.
%
%      H = UIJ2 returns the handle to a new UIJ2 or the handle to
%      the existing singleton*.
%
%      UIJ2('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in UIJ2.M with the given input arguments.
%
%      UIJ2('Property','Value',...) creates a new UIJ2 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before uij2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to uij2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help uij2

% Last Modified by GUIDE v2.5 03-Oct-2011 20:27:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @uij2_OpeningFcn, ...
                   'gui_OutputFcn',  @uij2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before uij2 is made visible.
FUNCIÓN uij2_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to uij2 (see VARARGIN)

% Choose default command line output for uij2
handles.output = hObject;

%generacion de las condiciones iniciales
q=evalin('base', 'q');
i=2;
for i=1:2
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text1, 'string', ['theta1=']);
            else
                set(handles.text1, 'string', ['d1=']);
        end
    else
        if q(i)==['theta',num2str(i)]
                set(handles.text2, 'string', ['theta2=']);
            else
                set(handles.text2, 'string', ['d2=']);
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes uij2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = uij2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
```

```
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ui
uii=ui;
q=evalin('base','q');
uii=evalin('base','uij(DH,q)');

a=str2double(get(handles.edit1, 'string'));
b=str2double(get(handles.edit2, 'string'));
uu=sym([]);
i=2;
j=2;
for i=1:2
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            eval(['theta',num2str(i),'=AA']);
        else
            AA=a;
            eval(['d',num2str(i),'=AA']);
        end
    else
        if q(i)==['theta',num2str(i)]
            BB=b;
            eval(['theta',num2str(i),'=BB']);
        else
            BB=b;
            eval(['d',num2str(i),'=BB']);
        end
    end
end
```

```
for i=1:2
    for j=1:2
        if i==1
            if j==1
                uu=uii(:,:,i,j);
                uu=eval(uu);
                set(handles.text3, 'string', num2str(uu,'%3.1f    '));
                set(handles.text7, 'string',['U11=']);
            else
                uu=uii(:,:,i,j);
                uu=eval(uu);
                set(handles.text4, 'string', num2str(uu,'%3.1f    '));
                set(handles.text9, 'string',['U12=']);
            end
        else if j==1
                uu=uii(:,:,i,j);
                uu=eval(uu);
                set(handles.text5, 'string', num2str(uu,'%3.1f    '));
                set(handles.text8, 'string',['U21=']);
            else
                uu=uii(:,:,i,j);
                uu=eval(uu);
                set(handles.text6, 'string', num2str(uu,'%3.1f    '));
                set(handles.text10, 'string',['U22=']);
            end
        end
    end
end


FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


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

```
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
return
else
winopen(strcat(Path,MANUAL));
end
```

APENDICE B.2.17

Script uij3.m

```
FUNCIÓN varargout = uij3(varargin)
% UIJ3 M-file for uij3.fig
%      UIJ3, by itself, creates a new UIJ3 or raises the existing
%      singleton*.
%
%      H = UIJ3 returns the handle to a new UIJ3 or the handle to
%      the existing singleton*.
%
%      UIJ3('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in UIJ3.M with the given input arguments.
%
%      UIJ3('Property','Value',...) creates a new UIJ3 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before uij3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to uij3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help uij3

% Last Modified by GUIDE v2.5 01-Oct-2011 11:14:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @uij3_OpeningFcn, ...
                   'gui_OutputFcn',  @uij3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before uij3 is made visible.
```

```
FUNCIÓN uij3_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to uij3 (see VARARGIN)

% Choose default command line output for uij3
handles.output = hObject;

%generacion de cuales son las condiciones iniciales
q=evalin('base', 'q');
i=3;
for i=1:3
    if i==1
        if q(i)==['theta',num2str(i)]
                set(handles.text1, 'string', ['theta1=']);
            else
                set(handles.text1, 'string', ['d1=']);
        end
    else if i==2
            if q(i)==['theta',num2str(i)]
                set(handles.text2, 'string', ['theta2=']);
            else
                set(handles.text2, 'string', ['d2=']);
            end
        end
        if q(i)==['theta',num2str(i)]
                set(handles.text15, 'string', ['theta3=']);
            else
                set(handles.text15, 'string', ['d3=']);
        end
    end
end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes uij3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = uij3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in pushbutton2.
FUNCIÓN pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ui
uii=ui;
q=evalin('base','q');
uii=evalin('base','uij(DH,q)');

a=str2double(get(handles.edit1, 'string'));
b=str2double(get(handles.edit2, 'string'));
c=str2double(get(handles.edit3, 'string'));
uu=sym([]);
i=3;
j=3;
for i=1:3
    if i==1
        if q(i)==['theta',num2str(i)]
            AA=a;
            eval(['theta',num2str(i),'=AA']);
        else
            AA=a;
            eval(['d',num2str(i),'=AA']);
        end
    else if i==2
            if q(i)==['theta',num2str(i)]
                BB=b;
                eval(['theta',num2str(i),'=BB']);
            else
                BB=b;
                eval(['d',num2str(i),'=BB']);
            end
        end
        if q(i)==['theta',num2str(i)]
                CC=c;
                eval(['theta',num2str(i),'=CC']);
        else
                CC=c;
                eval(['d',num2str(i),'=CC']);
```

```
        end
    end

end

for i=1:3
    for j=1:3
        if i==1
            if j==1
                uu=uii(:,:,i,j);
                uu=eval(uu);
                set(handles.text3, 'string', num2str(uu,'%3.1f    '));
                set(handles.text7, 'string',['U11=']);
            else if j==2
                    uu=uii(:,:,i,j);
                    uu=eval(uu);
                    set(handles.text4, 'string', num2str(uu,'%3.1f
'));
                    set(handles.text9, 'string',['U12=']);
                else
                    uu=uii(:,:,i,j);
                    uu=eval(uu);
                    set(handles.text11, 'string', num2str(uu,'%3.1f
'));
                    set(handles.text13, 'string',['U13=']);
                end
            end
        else  if i==2
                if j==1
                    uu=uii(:,:,i,j);
                    uu=eval(uu);
                    set(handles.text5, 'string', num2str(uu,'%3.1f
'));
                    set(handles.text8, 'string',['U21=']);
                else if j==2
                        uu=uii(:,:,i,j);
                        uu=eval(uu);
                        set(handles.text6, 'string', num2str(uu,'%3.1f
'));
                        set(handles.text10, 'string',['U22=']);
                    else
                        uu=uii(:,:,i,j);
                        uu=eval(uu);
                        set(handles.text12, 'string', num2str(uu,'%3.1f
'));
                        set(handles.text14, 'string',['U23=']);
                    end
                end
            else
                if j==1
                    uu=uii(:,:,i,j);
                    uu=eval(uu);
                    set(handles.text16, 'string', num2str(uu,'%3.1f
'));
                    set(handles.text18, 'string',['U31=']);
```

```
                else if j==2
                        uu=uii(:,:,i,j);
                        uu=eval(uu);
                        set(handles.text17, 'string', num2str(uu,'%3.1f
'));
                        set(handles.text19, 'string',['U32=']);
                    else
                        uu=uii(:,:,i,j);
                        uu=eval(uu);
                        set(handles.text20, 'string', num2str(uu,'%3.1f
'));
                        set(handles.text21, 'string',['U33='])
                    end
                end
            end
        end
    end
end

FUNCIÓN edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

FUNCIÓN edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
```

```
% --- Executes during object creation, after setting all properties.
FUNCIÓN edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



FUNCIÓN edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
FUNCIÓN edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to teoria (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()
```

```
% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()
```

```
% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
return
else
winopen(strcat(Path,MANUAL));
end
```

APENDICE B.2.18

Script vwagui.m

```
FUNCIÓN varargout = vwagui(varargin)
% VWAGUI M-file for vwagui.fig
%      VWAGUI, by itself, creates a new VWAGUI or raises the existing
%      singleton*.
%
%      H = VWAGUI returns the handle to a new VWAGUI or the handle to
%      the existing singleton*.
%
%      VWAGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      FUNCIÓN named CALLBACK in VWAGUI.M with the given input arguments.
%
%      VWAGUI('Property','Value',...) creates a new VWAGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before vwagui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to vwagui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help vwagui

% Last Modified by GUIDE v2.5 01-Oct-2011 11:37:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @vwagui_OpeningFcn, ...
                   'gui_OutputFcn',  @vwagui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before vwagui is made visible.
FUNCIÓN vwagui_OpeningFcn(hObject, eventdata, handles, varargin)
% This FUNCIÓN has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to vwagui (see VARARGIN)
global celdaCond

% Choose default command line output for vwagui
handles.output = hObject;
set(handles.uitable1,'Data',celdaCond);
set(handles.uitable1,'ColumnEditable',true(1:5));

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes vwagui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this FUNCIÓN are returned to the command line.
FUNCIÓN varargout = vwagui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
FUNCIÓN figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
tam=get(0,'ScreenSize');
pos=get(gcf,'Position');
xr=tam(3)-pos(3);
```

```
xp=round(xr/2);
yr=tam(4)-pos(4);
yp=round(yr/2);
set(gcf,'Position',[xp yp pos(3) pos(4)]);


% --- Executes on button press in siguiente.
FUNCIÓN siguiente_Callback(hObject, eventdata, handles)
% hObject    handle to siguiente (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cond=get(handles.uitable1,'Data');
dh=evalin('base','DH');
n=length(dh(:,1));
q=evalin('base','q');
for i=1:n
    qq=cell2mat(cond(i,1));
    vqq=cell2mat(cond(i,2));
    vvqq=cell2mat(cond(i,3));
    if q(i)==['theta',num2str(i)]
        A=qq;
        eval(['theta',num2str(i),'=A']);
        B=vqq;
        eval(['varTheta',num2str(i),'=B']);
        C=vvqq;
        eval(['vvarTheta',num2str(i),'=C']);
    else
        A=qq;
        eval(['d',num2str(i),'=A']);
        B=vqq;
        eval(['varD',num2str(i),'=B']);
        C=vvqq;
        eval(['vvarD',num2str(i),'=C']);
    end
end

evalin('base','nweu1');
ww=evalin('base','w');
aa=evalin('base','a');
varww=evalin('base','varw');
varvv=evalin('base','varv');

ww=eval(ww);
aa=eval(aa);
varww=eval(varww);
varvv=eval(varvv);
set(handles.text16, 'string', num2str(ww,'%5.1f    '));
set(handles.text15, 'string',['w=']);
set(handles.text18, 'string', num2str(aa,'%5.1f    '));
set(handles.text17, 'string', ['a=']);
set(handles.text10, 'string', num2str(varww,'%5.1f    '));
set(handles.text3, 'string',['varw=']);
set(handles.text14, 'string', num2str(varvv,'%5.1f    '));
set(handles.text13, 'string', ['varv=']);
```

```
% --------------------------------------------------------------------
FUNCIÓN teoria_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
FUNCIÓN introduccion_Callback(hObject, eventdata, handles)
% hObject    handle to introduccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_inicio()


% --------------------------------------------------------------------
FUNCIÓN nart_Callback(hObject, eventdata, handles)
% hObject    handle to nart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_numart()


% --------------------------------------------------------------------
FUNCIÓN dh_Callback(hObject, eventdata, handles)
% hObject    handle to dh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_DH()


% --------------------------------------------------------------------
FUNCIÓN com_Callback(hObject, eventdata, handles)
% hObject    handle to com (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_com()

% --------------------------------------------------------------------
FUNCIÓN cog_Callback(hObject, eventdata, handles)
% hObject    handle to cog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_cog()


% --------------------------------------------------------------------
FUNCIÓN modl_e_Callback(hObject, eventdata, handles)
% hObject    handle to modl_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()
```

```
% --------------------------------------------------------------------
FUNCIÓN modn_e_Callback(hObject, eventdata, handles)
% hObject    handle to modn_e (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_algoritmo()


% --------------------------------------------------------------------
FUNCIÓN acercade_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
acercade()


% --------------------------------------------------------------------
FUNCIÓN a_yuda1_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ayuda()


% --------------------------------------------------------------------
FUNCIÓN mph_Callback(hObject, eventdata, handles)
% hObject    handle to mph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_mph()


% --------------------------------------------------------------------
FUNCIÓN uij_Callback(hObject, eventdata, handles)
% hObject    handle to uij (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_uij()


% --------------------------------------------------------------------
FUNCIÓN ji_Callback(hObject, eventdata, handles)
% hObject    handle to ji (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_ji()


% --------------------------------------------------------------------
FUNCIÓN d_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

t_d()


% --------------------------------------------------------------------
FUNCIÓN h_Callback(hObject, eventdata, handles)
% hObject    handle to h (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_h()


% --------------------------------------------------------------------
FUNCIÓN c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_c()


% --------------------------------------------------------------------
FUNCIÓN rot_Callback(hObject, eventdata, handles)
% hObject    handle to rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_rot()


% --------------------------------------------------------------------
FUNCIÓN vwag_Callback(hObject, eventdata, handles)
% hObject    handle to vwag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t_vwag()


% --------------------------------------------------------------------
FUNCIÓN a_yuda_Callback(hObject, eventdata, handles)
% hObject    handle to a_yuda1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN modelos_Callback(hObject, eventdata, handles)
% hObject    handle to modelos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
FUNCIÓN funcion_Callback(hObject, eventdata, handles)
% hObject    handle to funcion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
FUNCIÓN abrir_pdf_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_pdf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[MANUAL Path]=uigetfile({'*.pdf'},'Abrir documento');
if isequal(MANUAL,0)
return
else
winopen(strcat(Path,MANUAL));
end
```