Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

“Interfaz gráfica para la identificación para plataforma de 2 grados de libertad”

Autor: Carlos Raúl Carballo Garaboto

Tutor: Dr. Ángel Ernesto Rubio Rodríguez

Ing. Pablo José Prieto Entenza

Santa Clara

2011

"Año 53 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

“Interfaz gráfica para la identificación para plataforma de 2 grados de libertad”

Autor: Carlos Raúl Carballo Garaboto

E-mail: ccarballo@uclv.edu.cu

Tutor(es): Dr. Ángel Ernesto Rubio Rodríguez

E-mail: erubio@uclv.edu.cu

Ing. Pablo José Prieto Entenza

E-mail: pablop@uclv.edu.cu, FIE

Santa Clara
2011
"Año 53 de la Revolución"

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____

Firma del Autor

_____

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Ayudar al que lo necesita no sólo es parte del deber, sino de la felicidad.

José Martí.

DEDICATORIA

A mi madre que es la persona más honesta, sencilla y comprensiva que jamás he conocido, por ser ella la fuente de todo mi ser y por estar siempre a mi lado.

A mi padre Juan Carlos por ser mi guía, dedicación y voluntad, por ser la persona más cariñosa que existe.

A mi hermanita Paula María por brindar su felicidad y cariño.

A mi hermano Alejandro por ser importante en mi vida.

A mi familia y amigos.

Gracias a todos por existir.

AGRADECIMIENTOS

A mis padres Xiomara y Juan Carlos, por estar a mi lado en las buenas y en las malas, por ser ellos motivos e inspiración de todo lo que he logrado en la vida.

A toda mi familia, por su gran apoyo.

A mis tutores Ángel Ernesto Rubio y Pablo Entenza Prieto, por la paciencia y apoyo que me han dado.

A todos mis profesores por haber hecho de mí una mejor persona y a Oscar por su incontable apoyo.

A Arnaldo y Yidier, por haberme ayudado muchísimo.

A mi amigo Arian por su incondicionalidad y su esfuerzo.

A Reinier (Zona) porque además de ser mí amigo es como un hermano para mí

A mis amigos del aula.

A todos que de una forma u otra hicieron posible que mi gran sueño se hiciera realidad.

RESUMEN

El presente trabajo de diploma se basa en la necesidad de diseñar una interfaz gráfica que permita realizar el proceso de identificación, que ayude de una manera más fácil a realizar dicho proceso en simuladores de conducción. El objetivo general es: desarrollar una interfaz amigable que permita la realización semiautomática del proceso de identificación del simulador de conducción de dos grados de libertad de SIMPRO. La interfaz se realiza para hacer cómodamente el proceso de identificación lo cual constituye un gran inconveniente para la producción masiva de los simuladores de SIMPRO.

La parte correspondiente al software de diseño se realizó en el lenguaje del Matlab 7.4, muy adecuado para la aplicación de tareas en tiempo real por sus prestaciones y alto alcance. La interfaz implementada fue probada a través de una simulación.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Un simulador es un sistema mecánico que permite la simulación de un proceso, donde se reproduce su comportamiento. Los simuladores reproducen sensaciones físicas (velocidad, aceleración, percepción del entorno) o el comportamiento de los equipos de la máquina que se pretende simular. Para simular las sensaciones físicas se puede recurrir a complejos mecanismos hidráulicos comandados por potentes ordenadores que mediante modelos matemáticos consiguen reproducir sensaciones de velocidad y aceleración.

Para simular el comportamiento de los equipos de la máquina simulada se pueden recurrir varias técnicas como son, elaborar un modelo de cada equipo, utilizar el equipo real o bien se puede utilizar el mismo software que corre en el equipo real pero haciéndolo correr en un ordenador más convencional (y por lo tanto más barato).

Entre los distintos tipos de simuladores se encuentran los simuladores de vuelo, de conducción, de carreras, de trenes, de vida, de negocios, de redes, clínico medico, entre otros. Los simuladores de conducción, además de ser usados para la capacitación y entrenamiento de potenciales conductores de vehículos, tienen una aplicación muy cercana y creciente en juegos y entretenimientos. Estos van teniendo mayor aceptación a nivel mundial en la medida de que logran recrear con mayor precisión y exactitud la inmersión en un mundo virtual y los más complejos recrean incluso las sensaciones de movimiento con plataformas móviles de varios grados de libertad (GDL). Dichas plataformas permiten el entrenamiento del personal y se minimiza así el uso de vehículos reales, con el consiguiente ahorro de recursos. Además evitan posibles accidentes de los principiantes en condiciones de peligrosidad, ya que el ordenador permite reproducir condiciones extremas de conducción en un ambiente virtual, seguro y totalmente controlado. En Cuba, da solución a las necesidades del país el Centro de Investigación y Desarrollo de Simuladores (SIMPRO), al desarrollar simuladores de conducción

ampliamente utilizados en entrenamiento de personal de la producción y los servicios, a la vez que en juegos virtuales (Entenza, 2009, Moreno, 2000).

Los análisis de factibilidad económica indican que la solución más económica para las plataformas de simuladores es la que utiliza actuadores neumáticos lineales, ya que estos sistemas tienen grandes ventajas como la limpieza, altas razones de carga contra peso y carga contra volumen, altas velocidades y grandes fuerzas, a través de un mecanismo operacional simple, lo que permite el desarrollo de manipuladores compactos, ligeros y rentables, que pueden ser utilizados en una gran variedad de aplicaciones (Moreno, 2000).

El Grupo de Automatización, Robótica y Percepción (GARP) del Departamento de Automática de la UCLV, en estrecha colaboración con SIMPRO, tiene la tarea de diseñar e implementar el sistema de control para las plataforma electro-neumática desarrolladas por dicha empresa. El control de estos sistemas es extremadamente complicado dada las no linealidades intrínsecas a la tecnología neumática.

Una de las ventajas del simulador de SIMPRO es la apuesta a la neumática para abaratar costos y simplificar mantenimientos pero su utilización da consigo el problema de control de la neumática.

Al grupo GARP se le da la tarea de resolver el problema de control desarrollándose una propuesta que parte de la identificación experimental del sistema, con el objetivo de estimar el modelo real de la planta para luego poder sintetizar un controlador robusto basado en el modelo obtenido mediante la identificación experimental (Rubio et al., 2009, Hernández et al., 2004). Actualmente el proceso de identificación y ajuste de los controladores se hace desde la línea de comando de Matlab y sólo es comprensible para especialistas con experiencia en el tema. Como no existe una interfaz amigable que permita hacer cómodamente este proceso lo cual constituye un gran inconveniente para la producción masiva de los simuladores de SIMPRO (Rubio, 2009).

De lo anterior se formula la siguiente **hipótesis** de trabajo, si se logra por Matlab el diseño de una interfaz amigable, se optimiza el correcto funcionamiento del proceso de identificación.

Por lo que se propone como **objetivo general** desarrollar una interfaz amigable que permita la realización semiautomática del proceso de identificación del simulador de conducción de SIMPRO.

Como **objetivos específicos** se establecen los siguientes:

1. Analizar críticamente las alternativas existentes para el desarrollo de la interfaz amigable.
2. Implementar una interfaz amigable que cumpla con los requerimientos necesarios.
3. Simular la interfaz desarrollada.

ORGANIZACIÓN DEL INFORME

El siguiente informe consta de tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. En el primer capítulo se comenta sobre los simuladores desarrollados en el mundo y sus actuadores, la importancia de los simuladores de conducción, la utilización de los simuladores de SIMPRO, así como sus características técnicas; además se realizó una comparación en la elección de una alternativa para el desarrollo de una interfaz grafica para el desarrollo del proceso de identificación.

En el capítulo 2 se definirán los requerimientos necesarios para llevar a cabo la realización de la interfaz, así como su diseño.

En el capítulo 3 Se harán las comprobaciones y simulación del software.

CAPÍTULO 1. ESTADO DEL ARTE DE LOS SIMULADORES

Antes de adentrarnos en el estudio de los simuladores de conducción para el desarrollo de la interfaz gráfica es necesario realizar una breve reseña histórica acerca de la evolución de los robots paralelos y abordar acerca de las características propias del simulador de conducción de SIMPRO.

1.1. Antecedentes.

Los primeros trabajos teóricos relacionados con las estructuras mecánicas paralelas aparecieron cuando los geómetras franceses e ingleses realizaron sus estudios sobre los poliedros y sus aplicaciones (López, 2008).

El primer robot paralelo del que se tiene conocimiento es una plataforma sobre la que estaban colocados los asientos de un teatro con el fin de introducir un movimiento que brindase una apariencia más real del espectáculo. Esta plataforma denominada “Amusement Device”, fue patentada en 1931 por J.E. Gwinnett (López, 2008),(Merlet, 2006).

En 1942 W.L.V. Pollard patenta un robot paralelo llamado “Position-Controlling Apparatus”; este mecanismo estaba destinado a pintar automóviles (López, 2008). El robot consistía en tres brazos de dos eslabones cada uno, unidos mediante juntas universales ver Figura 1.1. Los tres actuadores de la base comandaban la posición de la herramienta, mientras que la orientación era proporcionada por otros dos actuadores situados en la base que transmitían el movimiento a la herramienta mediante la rotación proporcionada a unos cables flexibles (López, 2008).

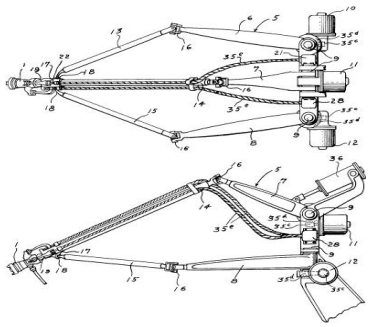


Figura 1.1.Robot paralelo patentado por W.L.V. Pollard.

Han sido otros mecanismos paralelos los que han logrado un mayor reconocimiento general y han contribuido a la aparición de un mayor número de publicaciones relacionadas con la robótica paralela. En 1947 V.E. Gough ideó un robot paralelo con seis actuadores lineales mostrado en la Figura 1.2, formando una estructura de octaedro (Merlet, 2006, Aracil et al., 2006). Este robot de seis grados de libertad fue utilizado en la empresa Dunlop para el ensayo de neumáticos de aviación y se presentó en un Congreso de La Federación Internacional de Sociedades de Ingenieros y Técnicos del Automóvil (FISITA) en 1962 (López, 2008).

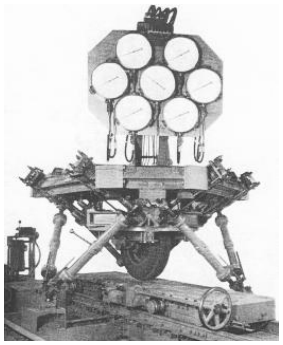


Figura 1.2.Robot paralelo de Gough

En la actualidad este ingenioso diseño constituye la base de la construcción de numerosos tipos de plataformas, entre ellas las famosas Mesas de Simulación Multiejes (MAST: Multi-Axis Simulation Table) muy empleadas para la simulación de conducción de todo tipo de vehículos, algunas variantes son mostradas en la Figura1.3 (Velazco, 2007).

Figura 1.3.Varios tipos de Mesas multiejes para la simulación de vehículos.

La estructura mecánica de la plataforma de Gough permite la realización de arquitecturas más rígidas usando cadenas cinemáticas idénticas. Ello implica poder manejar grandes cargas con una alta precisión además de reducciones en el costo y fácil mantenimiento (Velazco, 2007). Un ejemplo en una aplicación industrial de dicha plataforma se muestra en la Figura 1.4.

Figura1.4. Plataforma de Gough en una aplicación industrial.

Stewart presentó un artículo en el que describía una plataforma de 6 grados de libertad destinada a simular vuelos de avión (figura 1.5). Este constituye uno de los trabajos que hace un buen análisis académico de plataformas paralelas (Aracil et al., 2006).

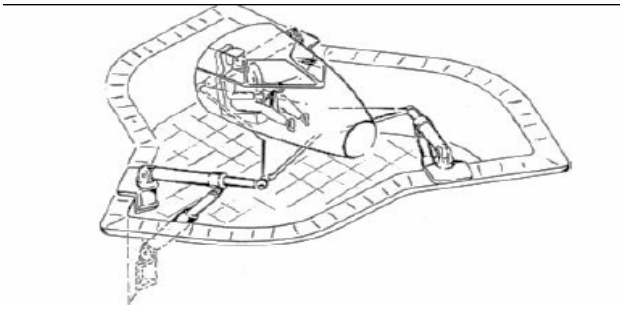


Figura 1.5.Esquema del diseño de la Plataforma de Stewart

Las máquinas cinemáticas paralelas (PKMs) desde los años 90 constituyen una opción de mucho interés que crece continuamente en diversos sectores de la industria como una herramienta útil, con prestaciones dinámicas potencialmente deseable, rápido tiempo de respuesta, rigidez, y exactitud aceptable (López, 2008). Entre los representantes de este tipo de robots paralelos se pueden citar el robot Delta , el Hexapods, el Hexagile y el robot Hermes (López, 2008). Dentro de sus principales aplicaciones están el desarrollo de herramientas de perforación, rebajado de piezas, soldadura, ensamblaje en la industria de automóviles, etc. Para el futuro se está valorando la posibilidad de implementar arquitecturas mixtas (serie-paralela) para el desarrollo de los robots (Velazco, 2007).

1.1.1. Tipos de Simuladores.

Hoy en día, muchas compañías están construyendo los simuladores del movimiento de la realidad virtual, no sólo para los aviones sino también para las naves, tren, carro que conduce el INRT; este sector es probablemente en el cual las estructuras paralelas son las más acertadas. Un ejemplo de tal simulador se presenta en la Figura1.6 (Merlet, 2006).

Figura 1.6. Simulador de vuelo

El simulador de conducción más grande es el simulador de conducción avanzado nacional (NADS) en la universidad del NADS de Iowa mostrado en la figura 1.7

Figura 1.7 . El NADS que conduce el simulador en la universidad de Iowa

Un simulador es Persival, (Figura 1.8): que tiene como objetivo el dar a los jinetes principiantes un entrenamiento preliminar sobre el caballo, ahora es disponible en el comercio PHS (Merlet, 2006).

Figura 1.8. El simulador Persival

Otros simuladores asombrosamente son el simulador de la bicicleta desarrollado por KAIST en Corea (Figura 1.9), y el sistema de Caren de Motek que se utilice para los deportes que entrenan y la rehabilitación médica (Merlet, 2006).

Figura 1.9. Simulador de la bicicleta por KAIST y el sistema de Caren de Motek

Por su aceptación a nivel mundial y su ampliamente desarrollo, los simuladores hoy en día son introducidos en el mercado para diferentes aplicaciones, según las prestaciones y requerimientos necesarios.

La alta exactitud de colocación y la rigidez de los robots paralelos permiten sus usos en varios campos industriales. Uno de los usos más acertados para las robustezas paralelas consiste en el empaquetado, cuya tarifa de transferencia es muy alta y la alta exactitud permite la manipulación de objetos frágiles, tales como

los que se resuelven en la industria alimenticia; también se han utilizado en usos misceláneos: prueba del calzado, decoración de la porcelana , fabricación de lentes .

1.1.2. Tipos de Actuadores.

Un mecanismo puede ser operado manualmente, o por medio de un dispositivo impulsor para generar el movimiento deseado. A estos dispositivos que se encargan de producir las fuerzas y/o torques para mover las estructuras mecánicas, se les denomina actuadores. En la robótica, los más usados son clasificados de acuerdo a los siguientes criterios (López, 2008) :

- Según el tipo de movimiento generado (Actuadores lineales).
- Según la naturaleza de la fuente primaria de energía (Actuadores neumáticos, hidráulicos y eléctricos).

Una clasificación de los diferentes tipos de actuadores anteriormente mencionados se muestra a continuación:

Actuadores neumáticos:

- Cilindros neumáticos
    - Simple efecto
    - Doble efecto
- Motores neumáticos

O Aletas rotativas

O Pistones axiales

Actuadores hidráulicos:

• Cilindro hidráulico

• Motor hidráulico

Actuadores eléctricos:

- Motores de corriente continua (DC)
    - Controlados por inducido
    - Controlados por excitación

- Motores de corriente alterna (AC)
    - Sincrónicos
    - Asincrónicos
- Motores paso a paso

Los actuadores hidráulicos permiten el manejo de una fuerza considerable, sus

aplicaciones se centran en el manejo de cientos de Newton-metros y la potencia de salida es de algunos Kilowatt (Krivts and Krejnin, 2006).

Los actuadores eléctricos generalmente se emplean cuando el movimiento requiere de un número de posiciones intermedias que se deben cambiar fácilmente, se caracterizan por la facilidad de control, sencillez y precisión.

Los actuadores neumáticos constituyen una tecnología que se ha venido introduciendo en los robots paralelos que requieren de un posicionamiento continuo, demostrando ser una tecnología barata, de respuesta rápida, elevada relación potencia-peso y fácil mantenimiento (Moore and Pu, 1996, Plattenburg, 2005, López, 2008).

1.2. Importancia de los simuladores de conducción.

Dichos simuladores tienen gran importancia pues son una guía o mecanismo para el aprendizaje de potenciales conductores de vehículos, tienen una aplicación muy cercana y creciente en juegos y entretenimientos. Estos permiten que se minimice de manera real la utilización de los vehículos y además contribuyen a la formación de los conductores. Además evitan posibles accidentes de los principiantes en condiciones de peligrosidad, ya que el ordenador permite reproducir condiciones extremas de conducción en un ambiente virtual, seguro y totalmente controlado. Además evitan posibles accidentes de los principiantes en condiciones de peligrosidad, ya que el ordenador permite reproducir condiciones extremas de conducción en un ambiente virtual, seguro y totalmente controlado (Aracil et al., 2006).

1.3. Simuladores de conducción de SIMPRO.

El simulador de conducción de sello SIMPRO es ampliamente utilizado en el entrenamiento de personal. Dicho simulador está constituido por una cabina de conducción que tiene consigo mandos reales que simulan el estado al cual se enfrenta el chofer del vehículo y un monitor, por el cual el conductor a través de la simulación puede observar el entorno virtual donde el interacciona.

Las simulaciones pueden llevarse a cabo tanto para vehículos pesados como para ligeros. La plataforma presenta un diseño formado por una estructura paralela, con cinco uniones universales y dos articulaciones constituidas por cilindros neumáticos, que son los encargados de darle al conductor la sensación de ladeo y cabeceo, la cual se muestra en la Figura 1.10 (Rubio, 2009).

Figura 1.10. Simulador de conducción.

La cabina pivotea sobre una columna central mediante una articulación pasiva en cuyo extremo superior se encuentra una unión universal. Los movimientos de la plataforma móvil se logran mediante la acción de los dos cilindros neumáticos (actuadores) que constituyen articulaciones actuadas cuyos desplazamientos lineales le imprimen al efector final rotaciones sobre dos ejes perpendiculares entre sí. Estas rotaciones simulan las pendientes del mundo virtual las cuales son visualizadas en el monitor ubicado en la propia cabina (Velazco, 2007).

La plataforma de SIMPRO es un robot paralelo de dos grados de libertad, donde la acción de cada uno de los actuadores tiene efecto sobre una sola articulación lo

que permite que esta se comporte de forma desacoplada. En ambos extremos de las articulaciones se encuentran uniones universales que le permiten los grados de movilidad necesarios para lograr las orientaciones de la plataforma móvil superior como se muestra en la Figura 1.11 (Entenza, 2009).

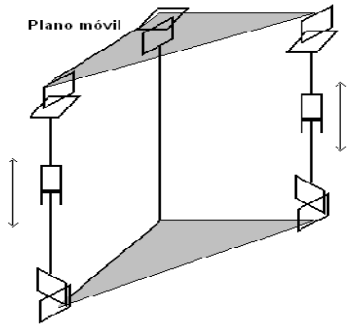


Figura 1.11. Arquitectura de la plataforma.

1.3.1. Características técnicas de las plataformas de los simuladores de SIMPRO.

Dichos simuladores según el fabricante del simulador, la condición crítica de movimiento a experimentar por el vehículo simulado se considera como la caída libre de un extremo pivoteando en el otro. Bajo estas condiciones, la plataforma debe ser capaz de alcanzar aceleraciones angulares de hasta 2 rad/seg2 que para los cilindros representa aceleraciones lineales de hasta 1000 mm/seg2 siendo esto muy importante para su uso y manejo (Rubio, 2009).

Los datos más importantes de la plataforma del simulador se ofrecen en la siguiente Tabla 1.1 (Moreno, 2000). El origen de coordenadas para las medidas de longitud y ubicación del centro de masa (CM) se establece en el pivote central. En ambos extremos de las articulaciones se encuentran uniones universales que le permiten los grados de movilidad necesarios para lograr las orientaciones de la plataforma móvil superior. Cada articulación electro-neumática está formada por un cilindro FESTO DNC-100-400 gobernado por una válvula proporcional de flujo FESTO MPYE-5-3/8 (Rubio, 2009).

Tabla 1.1

Parámetros	Valor
Masa total de la cabina	510 Kg
Posición del centro de masa en Z	480 mm
Posición del centro de masa en X	100 mm
Posición del centro de masa en Y	60 mm
Distancia del origen de cada cilindro	560 mm
Elongación de los cilindros	± 150 mm
Ángulo de ladeo y cabeceo	± 0.26 rad

1.4. Análisis para el desarrollo de la interfaz.

Existen diferentes alternativas para llevar a cabo el desarrollo de la interfaz gráfica necesaria para el proceso de identificación y control, entre ellas se encuentran: la realización a través de Visual C++, Labview y Matlab. Pero esto no significa que todas son fáciles de implementar y a su vez no posean inconvenientes, por lo que es necesario llegar a una conclusión determinante de cuál de ellas se va a implementar.

1.4.1. Elección del Software.

Para seleccionar dicho software es necesario realizar una comparación para analizar ventajas y desventajas y así ver que cumplan con los requerimientos necesarios.

1.4.1.1. Visual C++

Microsoft Visual Studio C++ es un entorno de desarrollo integrado (IDE) para lenguajes de programación C, C++ y C#. Esta especialmente diseñado para el desarrollo y depuración de código escrito para las API's de Microsoft Windows DirectX y la tecnología Microsoft .NET Framework. Permite la creación de proyectos estructurando de forma sencilla programas que incluyen muchos ficheros. Además, incluye las MFC (Microsoft Foundation Classes), equivalentes a las OWL (Object Window Library) de Borland. Estas clases facilitan la programación Windows, sin tener que utilizar directamente el API de Windows, ya que agrupan las librerías de Windows en clases C++. MFC es una jerarquía de clases C++ para programar en Windows, entre las cuales hay algunas de alto nivel que proporcionan funcionalidad general (por ejemplo la clase CWnd) y otras que implementan funciones más específicas (Kumar, 2010, Coplien et al., 1995).

Además de esta jerarquía de clases, MFC da un modelo de desarrollo de aplicaciones llamado modelo documento/vista, que nos permite diseñar aplicaciones de forma que los datos de la aplicación vayan separados de los elementos que componen el interfaz de usuario. Esto permite modificar de forma independiente las dos partes del programa. Por tanto, las características principales son:

- Arquitectura documento/vista.
- Interfaz de documentos múltiples (MDI).
- Soporte para bases de datos.
- Programación para internet.
- Controles comunes para Windows 96/98 y NT.
- Soporte multithread.

Presenta bibliotecas y tiene como ventajas la no utilización de una plataforma de simulación pues ella misma genera un ejecutable, pero es difícil de implementar el tiempo real (Kumar, 2010).

1.4.1.2. Labview

Es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.

Este programa fue creado por Nationals Instruments (1976) para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986. Ahora está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux. Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control de todo tipo de electrónica (Instrumentación electrónica) sino también a su programación empotrada. Un lema tradicional de LabVIEW es: "La potencia está en el Software", que con la aparición de los sistemas multinúcleos se ha hecho aún más patente. Entre sus objetivos están el reducir el tiempo de desarrollo de aplicaciones de todo tipo (no sólo en ámbitos de Pruebas, Control y Diseño) y el permitir la entrada a la informática a profesionales de cualquier otro campo. LabVIEW consigue combinarse con todo tipo de software y hardware, tanto del propio fabricante: tarjetas de adquisición de datos, PAC, Visión, instrumentos y otro Hardware como de otros fabricantes (Instruments, 2010).

Es usado principalmente por ingenieros y científicos para tareas como:

- Adquisición de datos y análisis matemático
- Comunicación y control de instrumentos de cualquier fabricante
- Automatización industrial y programación de PACs (Controlador de Automatización Programable)
- Diseño de controladores: simulación, prototipaje rápido, hardware-en-el-ciclo (HIL) y validación
- Control y supervisión de procesos
- Visión artificial y control de movimiento

- Robótica
- Domótica y redes de sensores inalámbricos
- En 2008 el programa fue utilizado para controlar el LHC, el acelerador de partículas más grande construido hasta la fecha.
- Pero también juguetes como el Lego Mindstorms o el WeDo lo utilizan, llevando la programación gráfica a niños de todas las edades.

Su principal característica es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación pueden hacer (programas) relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. También es muy rápido hacer programas con Labview y cualquier programador, por experimentado que sea, puede beneficiarse de él. Incluso existen buenas prácticas de programación para optimizar el rendimiento y la calidad de la programación (Instruments, 2010).

Presenta facilidades para el manejo de:

- Interfaces de comunicaciones:
    - Puerto serie
    - Puerto paralelo
    - TCP/IP, UDP, DataSocket
    - Bluetooth
    - USB
- Capacidad de interactuar con otros lenguajes y aplicaciones:
    - DLL: librerías de funciones
    - .NET
    - ActiveX
    - Multisim
    - Matlab/Simulink

- AutoCAD, SolidWorks, etc

- Herramientas gráficas y textuales para el procesado digital de señales.
- Visualización y manejo de gráficas con datos dinámicos.
- Adquisición y tratamiento de imágenes.
- Control de movimiento (combinado incluso con todo lo anterior).
- Tiempo Real estrictamente hablando.
- Programación de FPGAs para control o validación.
- Sincronización entre dispositivos.

A pesar de tener grandes prestaciones presenta problemas para implementar el procesamiento de datos referido al proceso de obtención y validación del modelo. También al igual que C++ genera un ejecutable pero necesita un núcleo mínimo del software base (Instruments, 2010).

1.4.1.3. Matlab

Es un lenguaje de alto funcionamiento para computación técnica, Este integra computación, visualización, y programación, en un entorno fácil de usar donde los problemas y las soluciones son expresados en la más familiar notación matemática. Los usos más familiares de Matlab son:

- Matemática y Computación.
- Desarrollo de algoritmos.
- Modelado , simulación.
- Análisis de datos, exploración y visualización.
- Gráficas científicas e ingenieriles.
- Desarrollo de aplicaciones, incluyendo construcción de interfaces gráficas de usuario.

MATLAB es un sistema interactivo cuyo elemento básico de almacenamiento de información es la matriz, que tiene una característica fundamental y es que no necesita dimensionamiento. Esto le permite resolver varios problemas de computación técnica (especialmente aquellos que tienen formulaciones matriciales

y vectoriales) en una fracción de tiempo similar al que se gastaría cuando se escribe un programa en un lenguaje no interactivo como C o FORTRAN.

Matlab presenta una familia de soluciones a aplicaciones específicas de acoplamiento rápido llamadas ToolBoxes. Los toolboxes son colecciones muy comprensibles de funciones MATLAB, o archivos de matlab (M-files) que extienden el entorno de MATLAB para resolver clases particulares de problemas, 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 (MathWorks, 2007a).

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. Dicho software es muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL (MathWorks, 2007b).

Entre sus principales **características** se encuentran:

- Cálculo numérico rápido y con alta precisión.
- Manejo simbólico.
- Visualización avanzada.
- Programación mediante un lenguaje de alto nivel.
- Programación estructurada y orientada a objetos.

- Soporte básico para diseño de interfaz gráfica.
- Extensa biblioteca de funciones.
- Aplicaciones especializadas para algunas ramas de ciencias e ingeniería (toolboxes).

Operación:

- Simple y eficiente.
- Interactivo.
- Sistema de ayuda en línea.
- Interacción con otros entornos.

Es la herramienta matemática más completa debido a su amplia aplicación; pero tiene como desventajas el hecho de necesitar plataforma de software para llevar a cabo la simulación.

1.5. Conclusiones Parciales

- La importancia que hoy en día tienen los simuladores de conducción constituye una fuerte premisa para realizar el proceso de identificación, para así contribuir al desarrollo de los mismos.

- Se realiza una comparación entre las distintas alternativas de software para la implementación de la interfaz grafica que permita llevar a cabo la identificación de una forma más sencilla, para así contribuir al desarrollo de estos simuladores de conducción.

Por todo lo anterior planteado se llega a la conclusión de que es necesario en la realización de una interfaz gráfica para la identificación y control, la herramienta de MATLAB.

CAPÍTULO 2. DESARROLLO DE INTERFACES GRÁFICAS

Como se trata en el capítulo anterior nuestra terea consiste en poder realizar el proceso de identificación mediante una interfaz gráfica que le permita al usuario de una forma más sencilla llevar a cabo dicha operación. Para el desarrollo de esa interfaz se llega a la conclusión que el software donde se implementa es el Matlab, pero es necesario tener en cuenta una serie de requerimientos que permita la exactitud en la respuesta deseada.

2.1. Definición de Requerimientos.

2.1.1. Generación de la PRBS.

Para poder conocer el modelo real de la planta es necesario realizar el proceso de identificación, pero por la complejidad de determinar analíticamente muchos de los parámetros de los modelos, se recomienda determinar el modelo por métodos de identificación experimental (Rubio et al., 2007a, Rubio et al., 2007b, Rubio et al., 2007c, Rubio et al., 2001, Rubio et al., 2005, Rubio, 2009).

La determinación práctica de algunos de los parámetros que conforman los modelos matemáticos desarrollados para los actuadores electro-neumáticos, no es muy sencilla. Especialmente complicados son los relacionados con las fricciones y el flujo a través de la válvula, más si se considera que las válvulas son realmente subdimensionadas. Es por eso que es una necesidad recurrir a técnicas de identificación experimental para obtener el modelo del sistema con el cual diseñar los controladores (Rubio, 2009). Uno de los autores que recurre a esta técnica es Varseveld que identifica el sistema en lazo abierto con una señal PRBS (Pseudo Random Binary Sequence), de diferentes amplitudes y obtiene los modelos para las posiciones centro y extremos del cilindro, donde se corrobora la variación de la razón de amortiguamiento del sistema en función de la posición. Para el diseño de la estrategia de control se escoge el modelo de la posición central (Rubio, 2009, Varseveld and Bone, 1997).

Como la dinámica del sistema es claramente dependiente de la posición de la carga y el sistema es de tipo uno, resulta conveniente identificarlo en lazo cerrado alrededor de diferentes posiciones (Rubio, 2009).

Tener en cuenta el subdimensionamiento de la válvula y no promediar las constantes de tiempo de las cámaras, da como resultado un modelo que describe de forma más precisa la verdadera dinámica de los actuadores electro-neumáticos. Lo anterior se corrobora con la identificación experimental del sistema (Rubio, 2009).

La identificación de sistemas dinámicos se realiza excitando el sistema con algún tipo de señal y en donde se procesa la entrada y salida en determinado intervalo de tiempo, esta señal va a tener una excitación persistente.

La/s entrada/s al sistema deben ser cuidadosamente elegidas de forma que los datos recogidos proporcionen toda la información posible sobre el sistema. A este respecto, conviene tener en cuenta los siguientes aspectos:

- La señal de entrada debe contener el mayor número de frecuencias posibles. Por ejemplo, una señal senoidal pura no es adecuada en un experimento de identificación, puesto que sólo se obtendrá la respuesta del sistema para la frecuencia de dicha señal. Por el contrario, las señales escalonadas (con cambios bruscos) son muy utilizadas, puesto que contienen un espectro suficientemente amplio de frecuencias (Ljung, 1999).
- Para sistemas lineales, basta con utilizar dos niveles de entrada, preferiblemente barriendo todo el rango de variación permitido. En este tipo de sistemas se suelen utilizar señales binarias de duración aleatoria (conocidas como señales binarias aleatorias o pseudoaleatorias). Sin embargo, para sistemas no lineales es necesario trabajar con más de dos niveles de entrada (Ljung, 1999).

Por tanto para realizar el proceso de identificación el sistema va a ser excitado con una señal binaria persistente PRBS, donde luego se registrará la señal obtenida a

la salida y con ambas, aplicando métodos de estimación de parámetros, determinar el modelo dinámico discreto que las correlacione adecuadamente.

Este modelo debe ser validado con diferentes técnicas de análisis estadísticos, y de no ser el más adecuado debe cambiarse la estructura del modelo, el método de estimación o repetirse el experimento con otra señal de estimación hasta obtener un resultado satisfactorio (Ljung, 1999). Un estudio detallado de los métodos de estimación de parámetros puede verse en los libros de Ljung y Aguado (Ljung, 1999, Aguado, 2000).

2.1.2. Obtención del modelo paramétrico por diversos métodos.

Los modelos paramétricos quedan descritos mediante una estructura y un número finito de parámetros que relacionan las señales de interés del sistema (entradas, salida y perturbaciones). En muchas ocasiones es necesario realizar la identificación de un sistema del cuál no se tiene ningún tipo de conocimiento previo. En estos casos, se suele recurrir a modelos estándar, cuya validez para un amplio rango de sistemas dinámicos ha sido comprobada experimentalmente. Generalmente estos modelos permiten describir el comportamiento de cualquier sistema lineal. La dificultad radica en la elección del tipo de modelo (orden del mismo, número de parámetros, etc.) que se ajuste satisfactoriamente a los datos de entrada - salida obtenidos experimentalmente.

Los métodos de estimación de parámetros dan como resultado una función de transferencia en tiempo discreto que responda a:

$$y(k) = G(z)u(z) + H(z)e(k) \qquad \text{Ec.1.1}$$

A partir de la ecuación Ec.1.1 se definen diferentes estructuras según la forma en que se modele el ruido, con sus peculiaridades en cuanto al algoritmo que se utiliza para la determinación de G(z) y H(z) y las propiedades que sean asumidas para el ruido. Entre dichas estructuras se encuentran las siguientes:

ARX (Auto Regressive and Exogenous Variable): Estructura auto-regresiva [A(z).y(k)] con variables exógenas [B(z).u(k)], suponiendo un ruido blanco de

media cero y varianza constante. Se resuelve directamente por el algoritmo de mínimos cuadrados, que responda a la forma:

$$A(z)y(k) = B(z)u(k - nk) + e(k) \qquad \text{Ec.1.2}$$

ARMAX (Auto Regressive Moving Average and Exogenous Variable): Estructura auto-regresiva con variables exógenas, suponiendo un ruido blanco de media cero y varianza constante afectado por un filtro de media móvil. Se resuelve minimizando el error de predicción de forma iterativa, aplicando el algoritmo de mínimos cuadrados extendido. Respondiendo a la forma:

$$A(z)y(k) = B(z)u(k - nk) + C(z)e(k) \qquad \text{Ec.1.3}$$

OE (Output Error): Estructura auto-regresiva con variables exógenas, que sólo afecta a la relación entrada-salida (no perturbada) con un ruido blanco aditivo. Se resuelve con un algoritmo similar al ARMAX modificando el cálculo del error de predicción y el gradiente. Respondiendo a la forma:

$$y(k) = \frac{B(z)}{F(z)}u(n - nk) + e(k) \qquad \text{Ec.1.4}$$

BJ (Box-Jenkins): Estructura auto-regresiva con variables exógenas, cuya parte determinista no tiene parámetros comunes con la estocástica. Se resuelve con un algoritmo similar al ARMAX modificando el cálculo del error de predicción y el gradiente. Respondiendo a la forma:

$$y(k) = \frac{B(z)}{F(z)}u(n - nk) + \frac{C(z)}{D(z)}e(k) \qquad \text{Ec.1.5}$$

En todos los casos k representa el instante de muestreo k-ésimo, nk representa el retardo puro que pueda tener el sistema, mientras A, B, C, F y D son polinomios en z de orden na, nb, nc, nf y nd respectivamente. Detalles de cómo calcularlos pueden consultarse en los textos recomendados (Ljung, 1999, Aguado, 2000).

La obtención de un modelo de este tipo se realiza en dos pasos:

1. Elección de la estructura del modelo
2. Obtención de los parámetros del modelo que ajustan la respuesta del mismo a los datos de entrada y salida obtenidos experimentalmente.

2.1.3. Pre-procesamiento de la señal medida.

Los datos registrados pueden tener deficiencias que implican efectos devastadores en el resto del proceso de identificación, como son las siguientes:

- Presencia de perturbaciones de alta frecuencia, por encima de las frecuencias de interés en la respuesta del sistema.
- Datos claramente erróneos, producidos por fallos en el hardware o software utilizados en el experimento de recogida de muestras.
- Desviaciones, desplazamientos o perturbaciones de baja frecuencia.

A continuación, se ve la forma de tratar cada una de estas deficiencias para conseguir datos adecuados para el proceso de identificación.

2.1.3.1. Eliminación de perturbaciones de alta frecuencia.

Estas perturbaciones se producen por fuentes de ruido ajenas al sistema y pueden ser evitadas mediante una correcta elección del período de muestreo. Si, tras el experimento, se observa que el período de muestreo escogido era innecesariamente pequeño (captándose por tanto estas perturbaciones indeseadas), se puede recurrir al diezmado de los datos, para evitar repetir el experimento con un período de muestreo mayor (Ljung, 1999).

2.1.3.2. Eliminación de datos erróneos.

Estos datos suelen presentarse de forma aislada, pero pueden tener un efecto muy negativo en el proceso de identificación. Por tanto, es fundamental eliminarlos antes de iniciar el proceso. Esto se realiza generalmente manualmente, eliminando dicho dato y aproximando su nuevo valor mediante interpolación. Para aplicaciones más avanzadas, existen algoritmos de detección de fallos que permiten corregir estos datos de forma casi automática (Ljung, 1999).

2.1.3.3. Tratamiento previo de los datos.

Antes de proceder a realizar la identificación es necesario analizar los datos registrados y decidir si son o no adecuados para el proceso de identificación, o si

necesitan algún tipo de tratamiento previo, como puede ser el filtrado de ruidos, eliminación de componentes de variación lenta u otros métodos (Ljung, 1999).

2.1.4. Validación del modelo.

En todo proceso de identificación es conveniente probar varias estructuras y diferentes órdenes dentro de cada estructura hasta dar con el modelo que mejor se ajuste a los datos obtenidos experimentalmente de la planta real. En definitiva, se trata de determinar cuándo un determinado modelo es lo suficientemente exacto para la aplicación requerida, proceso que se conoce habitualmente como validación del modelo (Ljung, 1999).

En general, la mayoría de los métodos de validación tratan de determinar si la respuesta del modelo se ajusta con suficiente exactitud a los datos de entrada-salida obtenidos mediante experimentación. A continuación se exponen algunos criterios típicos a la hora de descartar o elegir unos modelos respecto a otros.

- Validación en base a la aplicación del modelo:

Puesto que en la práctica es imposible determinar si un modelo responde exactamente al comportamiento de un sistema real, suele ser suficiente comprobar que el modelo es capaz de resolver el problema para el cual ha sido hallado (simulación, predicción, diseño de un controlador, etc.). Así, por ejemplo, si el controlador que ha sido ajustado por medio del modelo da buen resultado sobre el sistema real, se puede asegurar que el modelo era ‘válido’ para esta aplicación.

- Comprobación de parámetros físicos :

Para una determinada estructura que haya sido parametrizada en función de magnitudes físicas, un método importante de validación consiste en comparar el valor estimado de dichos parámetros y el que sería de esperar mediante el conocimiento previo que se tiene de la planta.

- Coherencia con el comportamiento de entrada-salida :

Para determinar si el comportamiento de entrada-salida está suficientemente caracterizado, puede ser necesario recurrir a diferentes métodos de identificación y comparar los resultados obtenidos. Por ejemplo, comparando los diagramas de

Bode de los modelos obtenidos mediante identificación paramétrica de diferentes estructuras, por el método de variables instrumentales y por análisis espectral, se puede determinar si la dinámica del sistema ha quedado suficientemente caracterizada (Aguado, 2000, Ljung, 1999).

- Reducción del modelo :

Un procedimiento para determinar si un modelo proporciona una descripción simple y apropiada de un sistema consiste en aplicarle algún método de reducción de modelos. Si una reducción en el orden del modelo no produce alteraciones apreciables en el comportamiento de entrada-salida del mismo, entonces el modelo original era innecesariamente complejo.

- Simulación :

Un procedimiento muy habitual que puede ser considerado como otra técnica de validación de modelos consiste en simular el modelo con un conjunto de entradas distintas a las utilizadas para identificación, y comparar la respuesta del modelo con la obtenida del sistema real.

- Análisis de residuos :

Se conocen como residuos de un sistema a los errores de predicción obtenidos según la expresión:

$$\varepsilon(t) = \varepsilon(t,\theta) = y(t) - y_e(t,\theta) \qquad \text{Ec.1.6}$$

Siendo θ el vector de parámetros del modelo, y(t) la respuesta real del sistema e y_e(t) la respuesta estimada por el modelo para la misma entrada.

Idealmente, estos residuos deben ser independientes de la entrada. Si no sucede así, significa que hay componentes en ε(t) que proceden de la entrada u(t), lo cual a su vez significa que el modelo no es capaz de describir completamente la dinámica del sistema.

Para realizar el estudio anterior, suele comprobarse la correlación entre el error de predicción y la entrada al sistema, según la expresión:

$$R_{\varepsilon u} = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t+\pi) * u(t) \qquad \text{Ec.1.7}$$

El modelo será tanto más exacto cuanto más se acerquen a cero los términos de la correlación anterior. Obviamente, el análisis de los residuos será un método de validación más eficaz si el conjunto de datos utilizados para realizar la correlación es distinto que el usado para la identificación del modelo (Ljung, 1999).

2.1.5. Síntesis del controlador.

En dependencia de las prestaciones que se necesiten, las estrategias de control basadas en un controlador PI son una opción válida para el posicionamiento de los actuadores electro-neumáticos. Estas estrategias brindan un desempeño adecuado, especialmente si se sintetizan a partir de un buen modelo de la planta y se complementan con técnicas no lineales para corregir fenómenos propios de estos sistemas. No son las más robustas, pero sí las más sencillas (Rubio, 2009).

Se propone como controlador el uso de un PI en cascada con un filtro de segundo orden para compensar los polos complejos conjugados de la planta.

En el trabajo de Karpenko (Karpenko and Sepehri, 2004), se sintetiza un controlador por QFT a partir de la variación paramétrica de la planta. Este controlador es un PI con un lazo interno de realimentación de presión, que garantiza el desempeño requerido en lazo cerrado ante las variaciones de los coeficientes y posibles perturbaciones. En él, la señal de referencia se pasa por un pre filtro que garantiza su suavidad evitando oscilaciones en la salida. Por su parte, Yamada propone un método de diseño de un controlador por ubicación de polos adaptable con linealizador neuronal que también realimenta presión (Yamada et al., 2000).

A partir de esto queda representada la planta identificada de la forma:

$$\mathrm{A(s)Y(s)} = \mathrm{B(s)U(s)} \qquad \text{Ec.1.8}$$

donde A y B son polinomios en s. Se propone un controlador de la forma:

$$\mathrm{R(s)U(s)} = \mathrm{S(s)(Y_d(s) - Y(s))} \qquad \text{Ec.1.9}$$

donde R y S son los polinomios del controlador y Y_d la dinámica de la señal de entrada o referencia. Luego, si la respuesta dinámica deseada Y_m ante la señal Y_d está descrita por:

$$A_m(s)Y_m(s) = B_m(s)Y_d(s) \qquad \text{Ec.1.10}$$

donde A_m y B_m son los polinomios del modelo en lazo cerrado deseado, es posible establecer que:

$$\frac{S(s)B(s)}{R(s)A(s)+S(s)B(s)} = \frac{S(s)B(s)}{A_c(s)} = \frac{B_m(s)}{A_m(s)} \qquad \text{Ec.1.11}$$

donde A_c es la ecuación característica del sistema:

$$A_c(s) = R(s)A(s) + S(s)B(s) \qquad \text{Ec.1.12}$$

Como:

$$A(s) = s(s^2 + a_1 s + a_0)$$

$$B(s) = k_n a_0 \qquad \text{Ec.1.13}$$

Mediante el proceso de modelado e identificación, se logra caracterizar la variación de los parámetros a_0 y a_1. Esta variación provoca que los polos complejos conjugados de A se desplacen por una región en dependencia de la posición del émbolo del cilindro. Para diseñar el controlador, se propone tomar el modelo medio del sistema formado por la media de cada coeficiente (Rubio, 2009):

$$\frac{B(s)}{A(s)} = \frac{k_{nm}\, a_{0m}}{s(s^2+a_{1m}+a_{0m})} \qquad \text{Ec.1.14}$$

La función de transferencia Ec.1.14 constituye la planta base para el diseño del controlador.

Se propone, como controlador S/R, un filtro que compensa los polos complejos de la planta, garantizando una adecuada estabilidad relativa, independientemente de las variaciones de los coeficientes (Janiszowski, 2004), y un controlador PI que

garantiza una adecuada respuesta del sistema ante perturbaciones (Karpenko and Sepehri, 2004). Quedando de la forma:

$$\frac{S(s)}{R(s)} = \frac{k_p(s+k_i)(s^2+a_{1m}s+a_{0m})}{s(s+\omega_a)^2} \qquad \text{Ec.1.15}$$

Los polos complejos conjugados del sistema se compensan con dos ceros aportados por S ubicados sobre los polos de A, por lo que es necesario introducir dos polos ω_a en R para mantener físicamente realizable el controlador. Asumiendo una compensación perfecta, el polinomio $(s^2 + a_{1m}s + a_{0m})$ es común en A y S, por lo que Ec.1.11 puede rescribirse como:

$$\frac{S'(s)B(s)}{(R(s)A'(s)+S'(s)B(s))} = \frac{S'(s)B(s)}{A_c'(s)} = \frac{B_m(s)}{A_m(s)} \qquad \text{Ec.1.16}$$

Donde A', S' y Ac' son los polinomios A, S y Ac sin incluir los polos complejos de la planta.

Luego, los polinomios R y S se obtienen resolviendo la ecuación diofántica correspondiente:

$$A_c'(s) = A_m(s) \qquad \text{Ec.1.17}$$

Con lo anterior, se tiene como ecuación característica del sistema, sin incluir el polinomio $(s^2 + a_{1m}s + a_{0m})$*:*

$$A_c'(s) = s^2(s+\omega_a)^2 + k_p a_{0m} k_{nm}(s+k_i) \qquad \text{Ec.1.18}$$

Como ecuación característica deseada, sin incluir el polinomio $(s^2 + a_{1m} + a_{0m})$, se propone:

$$A_m(s) = s(s+p_1)(s+p_2)(s^2 + 2\varphi\omega_n s + \omega^2) \qquad \text{Ec.1.19}$$

donde p1 y p2 serán dos polos reales no dominantes, mientras φ y ω_n serán, respectivamente, la razón de amortiguamiento y la frecuencia natural no amortiguada deseadas del sistema en lazo cerrado.

Imponiendo como condición de diseño que el controlador no amplifique las altas frecuencias, lo que provocaría en la práctica oscilaciones en la acción de control

perjudiciales para la válvula, se tiene una nueva ecuación a partir de hacer cero el límite de la magnitud de Ec.1.15 para $s = j\omega \to \infty$

$$2\log\left[\frac{{\omega_a}^2}{k_p k_i a_{0m}}\right] + 2\log\left[\frac{1}{{\omega_a}^2}\right] = \log\left[\frac{1}{{k_i}^2}\right] + \log\left[\frac{1}{{a_{0m}}^2}\right] \qquad \text{Ec.1.20}$$

Fijando la frecuencia natural no amortiguada y la razón de amortiguamiento, para ubicar los polos dominantes de lazo cerrado, se resuelve el sistema y se obtienen los parámetros del controlador Ec.1.15 (Rubio, 2009)

En realidad, los ceros del filtro compensador no cancelan exactamente los polos complejos de la planta, por lo que quedan en el numerador de la función de lazo abierto junto con el cero del PI. Mientras, en el denominador de lazo abierto permanecen los polos complejos del sistema, por lo que la ecuación característica real incluye un par de polos complejos más que A_m. De tal manera que la función de transferencia en lazo cerrado se ve afectada por la dinámica que aportan estos polos (Rubio, 2009).

Pero como los ceros del filtro compensador se ubican precisamente en el centro de la región por donde se desplazan los polos de la planta en función de la posición, en lazo cerrado estos últimos tenderán a los ceros compensadores con una ganancia muy pequeña multiplicando el factor exponencial que de ellos se deriva en la función inversa de Laplace. Por lo anterior, el aporte de estos polos complejos en la dinámica del sistema es despreciable (Rubio, 2009).

Como el sistema se identifica en lazo cerrado alrededor de la posición central del cilindro que es la dinámica más exigente, con la cual se obtiene un modelo donde a partir de este se puede ajustar el controlador propuesto donde se ubican los ceros compensadores corridos hacia la izquierda y separándolos un poco, buscando experimentalmente la mejor compensación a todo lo largo del cilindro.

2.1.6. Comunicación con la tarjeta de adquisición de datos.

Es necesario en la utilización como hardware de control una PC con una tarjeta de adquisición de datos HUMUSOFT MF614 y el Real Time Workshop de

MATLAB/SIMULINK (MathWorks, 2007b). Esta configuración brinda una gran potencia de cálculo y admite el control de ambas articulaciones con un período de muestreo de 1 ms, pero no es económicamente viable para la producción industrial de los Simuladores de Conducción. Con vistas a dar una solución económica, asociada a este trabajo se desarrolló una tesis de maestría cuya propuesta es el empleo de un controlador empotrado como hardware de control. En él, la estrategia de control propuesta brinda resultados satisfactorios (Machado, 2007, Machado et al., 2007).

La propuesta de Machado es emplear, como hardware de control, un controlador empotrado que corra en tiempo real el algoritmo de control propuesto en el presente trabajo, el cual cumple con los índices de funcionamiento requeridos por la aplicación. En este caso, el controlador empotrado no requiere lógica DSP debido a la relativa sencillez del controlador propuesto y la simplicidad de la plataforma, que sólo presenta dos grados de libertad (Machado, 2007).

De esta manera, la arquitectura de control Figura 2.1 queda como se describe a continuación: La PC recibe por el puerto paralelo las acciones reales que ejecuta el conductor en la cabina; calcula y visualiza los efectos dinámicos que tienen estas acciones en el mundo virtual y manda por puerto serie la orientación requerida por la cabina al controlador empotrado. El controlador recibe la orientación (ángulos de ladeo y cabeceo) y calcula la cinemática inversa para generar los valores deseados de elongación de los cilindros. Luego, en tiempo real, lee la posición de los actuadores a través de los canales A/D por donde se conectan potenciómetros lineales acoplados a los cilindros, calcula la acción de control a partir del algoritmo propuesto y la envía a las válvulas electro-neumáticas a través de los conversores D/A, cerrando así el lazo de control (Machado, 2007).

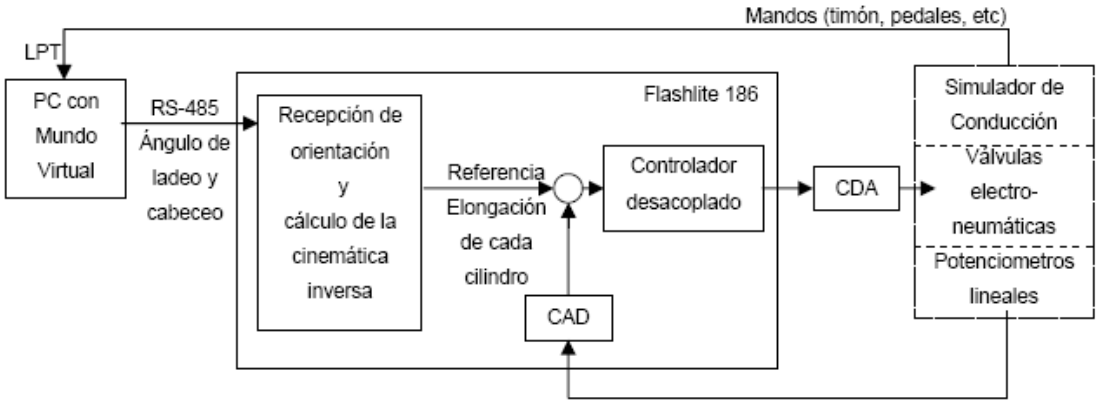


Figura 2.1 Arquitectura de control para los simuladores de conducción

2.2. Implementación de la interfaz grafica para la identificación.

En el capitulo anterior se realiza una comparación entre algunos software para utilizar el mas indicado, ya sea por sus prestaciones, dificultad o necesidad; concluyéndose a la utilización del Matlab, el cual es un software interactivo donde los usuarios pueden crear sus propias aplicaciones, ya sea a nivel industrial para resolver algún problema práctico o en centros de estudios y/o investigaciones donde se requiera la simulación de procesos, análisis y diseño de sistemas de control, procesamiento de señales, imágenes, etc. Además es empleado como software por excelencia para el control y simulación de robots paralelos según se reporta en la bibliografía consultada (Prattichizzo, 1998, Koekebakker, 2001).

Para llevar a cabo el proceso de identificación se implementó en el Matlab el algoritmo que se muestra en la Figura 2.3; la identificación experimental consiste en excitar el sistema mediante la aplicación de una señal de entrada y registrar la evolución de sus salidas durante un intervalo de tiempo. A partir de este vector entrada-salida, se procede a la estimación de los parámetros de un modelo, que mejor ajustan la respuesta del mismo a dicho vector.

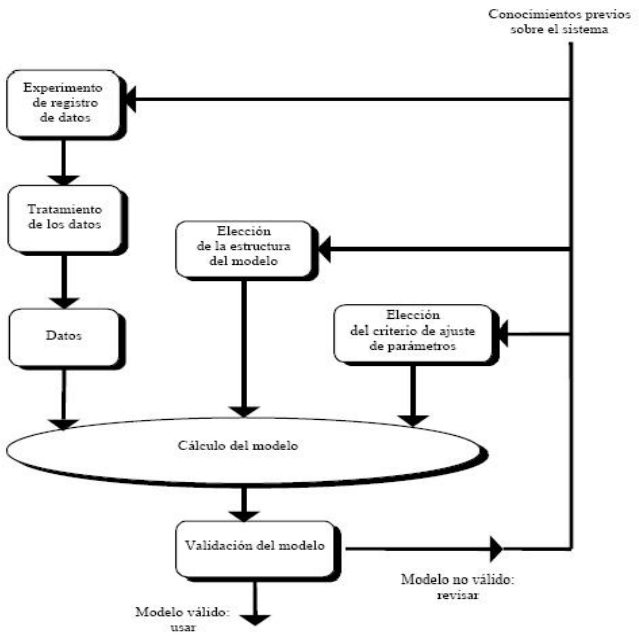


Figura 2.3 El proceso de identificación

El proceso de identificación comprende los siguientes pasos:

1. Obtención de datos de entrada – salida:

   Para ello se debe excitar el sistema mediante la aplicación de una señal de entrada y registrar la evolución de sus entradas y salidas durante un intervalo de tiempo.

2. Tratamiento previo de los datos registrados :

Los datos registrados están generalmente acompañados de ruidos indeseados u otro tipo de imperfecciones que puede ser necesario corregir antes de iniciar la identificación del modelo. Se trata, por tanto, de 'preparar' los datos para facilitar y mejorar el proceso de identificación.

3. Elección de la estructura del modelo:

Si el modelo que se desea obtener es un modelo paramétrico, el primer paso es determinar la estructura deseada para dicho modelo. Este punto se facilita en gran medida si se tiene un cierto conocimiento sobre las leyes físicas que rigen el proceso.

4. Obtención de los parámetros del modelo:

A continuación se procede a la estimación de los parámetros de la estructura que mejor ajustan la respuesta del modelo a los datos de entrada-salida obtenidos experimentalmente.

5. Validación del modelo:

El último paso consiste en determinar si el modelo obtenido satisface el grado de exactitud requerido para la aplicación en cuestión. Si se llega a la conclusión de que el modelo no es válido, se deben revisar los siguientes aspectos como posibles causas:

a) El conjunto de datos de entrada-salida no proporciona suficiente información sobre la dinámica del sistema.
b) La estructura escogida no es capaz de proporcionar una buena descripción del modelo.
c) El criterio de ajuste de parámetros seleccionado no es el más adecuado.

Cual sea la causa estimada, deberá repetirse el proceso de identificación desde el punto correspondiente. Por tanto, el proceso de identificación es un proceso iterativo, cuyos pasos pueden observarse en el organigrama de la figura anterior (Rubio et al., 2007c).

2.3. Desarrollo de la interfaz grafica.

El Matlab cuenta entre sus ventajas con un conjunto de herramientas denominadas Interfaz Gráfica de Usuarios (Graphical User Interface -GUI-), las mismas permiten mejorar el ambiente en el que se ejecuta una determinada aplicación programada en Matlab. Cuando se realiza un diseño con el editor de la GUI este crea un fichero con extensión .fig (Figura) y otro con extensión .m (Matlab).

- Un archivo .fig - Contiene la descripción de los componentes que contiene la interfaz.

- Un archivo .m - Contiene las funciones y los controles del GUI así como el callback

Un callback se define como la acción que llevará a cabo un objeto de la GUI cuando el usuario lo active.

2.3.1. Iniciando el GUIDE o GUI.

Para crear una GUI en Matlab usamos GUIDE, ya sea que tecleemos guide en la ventana de comandos de Matlab o lo ejecutemos del menú principal File.> New -> GUI como lo vemos en la Figura 2.4.

Una vez hecho lo anterior Matlab nos mostrará un área de diseño similar a la de la Figura 2.5. En la parte superior se encuentran los menús y opciones de GUIDE, en la parte izquierda se aprecian los diferentes controles y en la parte central el área de diseño donde pondremos los controles a usar.

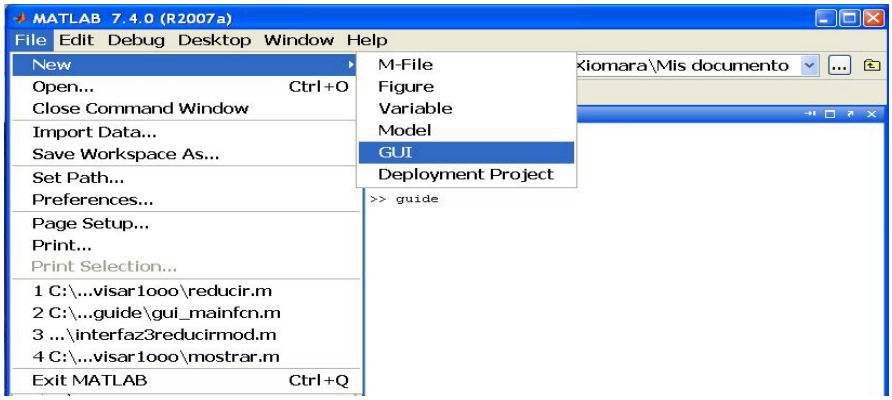


Figura 2.4. Ejecución de GUIDE usando el menú principal o la ventana de comandos

Figura 2.5. Área de trabajo de GUIDE

Para obtener la etiqueta de cada elemento de la paleta de componentes ejecutamos: File>>Preferentes y seleccionamos Show names in component palette.

Tenemos la siguiente presentación:

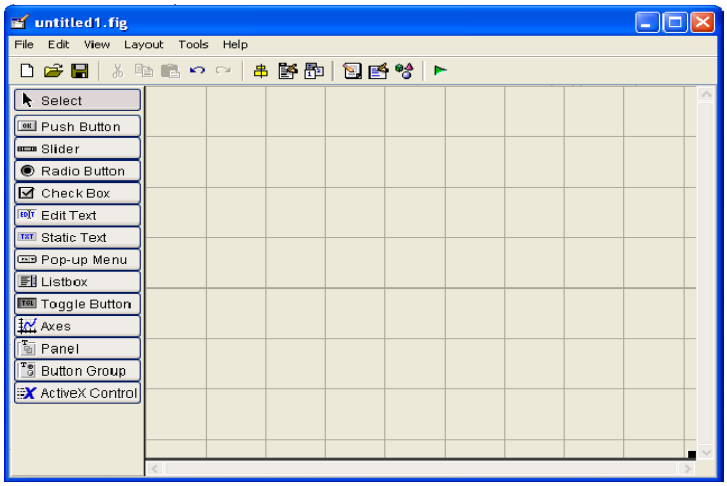


Figura 2.6. Entorno de diseño: componentes etiquetados.

2.3.2. Partes del GUI.

De los controles mostrados anteriormente mencionaremos las partes más importantes de GUIDE:

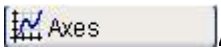
 Inspector de propiedades: Cada control cuenta con diferentes propiedades y es aquí donde podremos cambiar: el color, el nombre, el tag, el valor, el callback entre otros.

 Activar Figura: Una vez que hayamos terminado de diseñar presionamos este botón para activar la figura y poder probar nuestra GUI.

Push Button: Crea un botón.

Radio Button: Crea un botón circular.

Edit Text: Crea un campo de texto.



Axes: Crea una área para gráficas.

Panel: Crea un marco que puede contener otros controles.

Static Text: Crea un letrero.

Pop-up Menu: Crea un menú desplegable.

 Alinear objetos.

 Editor de orden de etiqueta.

 Editor del M-file.

 Editor de menú.

 Grabar y ejecutar (ctrl. + T).

Cada uno de los elementos de GUI, tiene un conjunto de opciones que podemos acceder como es mostrada en la Figura 2.7.

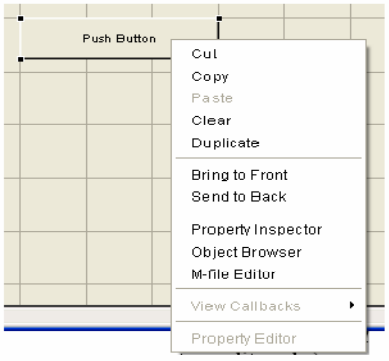


Figura 2.7. Opciones del componente.

La opción Property Inspector permite personalizar cada elemento ver Figura 2.8 y algunas de sus opciones mas importantes son mostradas en la Tabla 2.1.

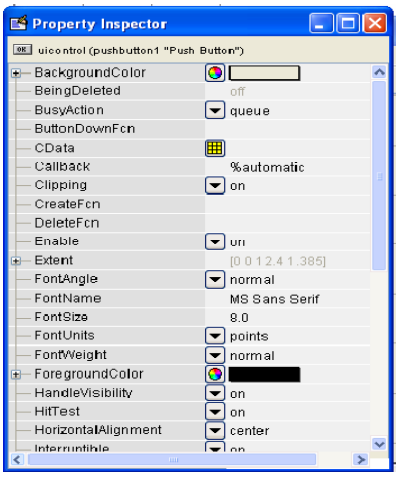


Figura 2.8. Entorno Property Inspector. Permite ver y editar las propiedades de un objeto.

Tabla 2.1

Property Inspector	
Background Color	Cambia el color del fondo del control.
CallBack	La propiedad más importante del control, ya que le dice al control que hacer cuando este se active.
Enable	Activa o desactiva un control.
String	En el caso de botones, cajas de texto,

	texto estático; es el texto que muestra el control.
Tag	Otra de las propiedades más importantes ya que con este es posible regresar datos o identificar al control.

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.

2.4. Primera Interfaz gráfica. Excitación del sistema.

Para poder realizar el proceso de identificación, como se vio anteriormente, es necesario excitar mi sistema para poder medir la salida y determinar su comportamiento. Teniendo esto en cuenta fue necesario realizar un estudio profundo de todas las herramientas con que cuenta el editor de interfaz del Matlab, así como las funciones que puede realizar cada uno de estos elementos en un determinado diseño.

2.4.1. Método para el desarrollo de la primera interfaz grafica.

Se accede al GUI como se describe anteriormente y abrimos un GUI en blanco en donde se realizará el diseño de nuestras interfaces gráficas. Como en el proceso de excitación es necesario el conocimiento de variables que son necesarias para determinar el comportamiento de la señal de excitación, en nuestro entorno de diseño (Figura 2.6) diseñamos la interfaz que es la encargada de realizar la excitación de mi sistema.

En primer lugar se diseña un panel en donde se encuentran las variables de la señal de excitación y las variables de la plataforma a la cual se excitará. Dentro de dicho panel se encontraran dos paneles, los cuales se llaman PRBS y

PLATAFORMA, las dos tienen sus variables propias, así de esta forma será más fácil para el usuario el manejo de dicha interfaz.

Como se habla anteriormente, para el proceso de identificación es necesario excitar la planta con una señal PRBS, por lo que es necesario en la generación de dicha señal un conjunto de variables que serán las encargadas de generar esa señal, que se comportara en dependencia de los valores que el usuario emplea para su creación.

Las variables que se encuentran en el panel (PRBS) son las siguientes:

- Punto de operación: no es más que la posición sobre el cual el sistema va a ser excitado para su posterior identificación.
- Amplitud: es la distancia donde se describirá el comportamiento de la PRBS.
- T est: es el tiempo de experimento estable.
- Texp: es el tiempo de experimento.
- Frec. máx. : frecuencia máxima de la PRBS

Luego de definir las variables para generar la PRBS se diseña un Push Button el cual se llama ‘Mostrar’, dicho control es el encargado de ejecutar a través del Callback de sí mismo un archivo punto m que no es más que el encargado de generar la PRBS. A continuación se muestra los códigos de implementación tanto para el ejecutable como para la PRBS:

Pushbutton_mostrar:

```
 disp ('Mostrar')
set (handles.edit_nombrefichero,'enable','off');
set (handles.pushbutton_save,'enable','off');

set (handles.edit_gain,'enable','on');
set (handles.edit_artic,'enable','on');
set (handles.pushbutton_ejecutar,'enable','on');

PO = handles.PO;
ampl = handles.ampl;
tEst = handles.tEst;
tExp = handles.tExp;
frecMax = handles.frecMax;
gain = handles.gain;

[t, u] = prbs( ampl, tEst, tExp, frecMax);
```

```
plot(t,u+PO);
save tmpfile PO ampl tEst tExp frecMax gain t u
```

PRBS:

```
function [t,u] = prbs( ampl, ts, tt, fmx)
tm = 0.001; % período de muestreo en seg
fm = 1/tm;  % frecuencia de muestreo en Hz

% Para 'PRBS': BAND = [0,B], donde 1/B es el número de muestras mínimo
que permanecerá cte la señal. Por tanto, el período mínimo de la PRBS
será 2*tm/B y la frecuencia máxima, que es la que se quisiera similar a
la de cruce del sistema, será fm*B/2 = fc De ahí que B se calcule como:
beta = 2*fmx/fm;
u = idinput(tt/tm,'PRBS',[0 beta],[-ampl/2 ampl/2]);
u = [zeros(ts/tm,1); u];
t = (0:length(u)-1)'*tm;
```

La señal PRBS es un vector formado por los valores de t y u. Estos vectores son salvados mediante un archivo punto mat llamado tmpfile como se puede ver anteriormente.

Luego de tener los datos de la PRBS y su valor, el siguiente paso es el que permitirá a cuál de las dos articulaciones se desea excitar para medir la salida correspondiente a la señal de entrada y luego conociendo la salida y la señal de entrada podemos estimar el modelo de la planta a través del proceso de identificación.

Para llevar a cabo la excitación de las articulaciones es necesario saber cuál va a ser excitada y para esto en la inicialización del programa se implementa lo siguiente:

```
if artsel == 1
        plot(artic1sign.signals(1).values)
    else
        plot(artic2sign.signals(1).values)
    end
```

Luego de tener elegida la articulación que va a ser excitada se ejecuta un mdl con la configuración del sistema en lazo cerrado mediante un punto m que a continuación se muestra:

Pushbutton_ejecutar:

```
disp('Ejecutar')
artsel = get(handles.edit_artic,'Value');
load tmpfile;
save tmpfile PO ampl tEst tExp frecMax gain t u artsel
Tsim = tEst + tExp;
```

```
Klc = gain;
T = 0.001;
pf = handles.filename;
open_system( pf )
delete(handles.figure1)
set_param(gcs,'Stop time',num2str(Tsim))
set_param([pf '/Selec'],'Value', num2str(artsel));
set_param([pf '/Constant'],'Value', num2str(PO));
set_param([pf '/Constant1'],'Value', num2str(PO));
set_param([pf '/Gain'],'Gain', num2str(Klc));
set_param([pf '/Gain1'],'Gain', num2str(Klc));
set_param(gcs,'ExtModeTrigMode','normal');
set_param(gcs,'ExtModeTrigDuration', Tsim/T);
set_param(gcs,'ExtModeArmWhenConnect','on');
set_param(gcs,'SimulationMode','external');
make_rtw;
set_param(gcs,'SimulationCommand','connect');
set_param(gcs,'SimulationCommand','start');
```

Configuración del mdl:

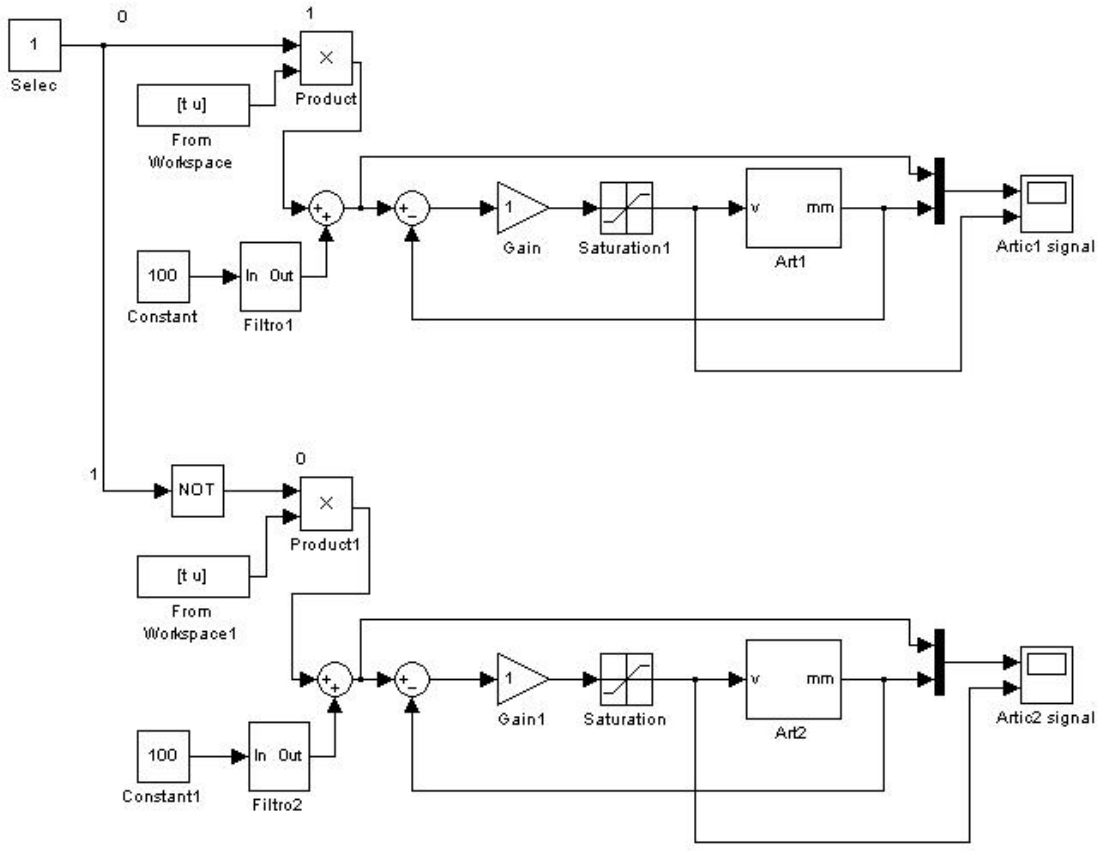


Figura 2.9. Configuración del mdl

Teniendo ya la PRBS y las articulaciones excitadas es necesario que se describan el comportamiento de la señal de entrada y las salidas, para eso se creó un Axes en el cuál se muestra dicho comportamiento.

Además se creó un control llamado 'Salvar' que se encarga de salvar los datos correspondiente a la simulación y un 'Edit Text' donde el usuario puede nombrar el archivo para proceder a su salvamento como se muestra a continuación:

Pushbutton_salvar:

```
T = 0.001; %período de muestreo
load tmpfile
Te = tEst;
Tn = tExp;
Kp = gain;

if artsel == 1
    data = artic1sign;
else
    data = artic2sign;
end
sp = data.signals(1).values(:,1);
pos = data.signals(1).values(:,2);
name = get(handles.edit_nombrefichero,'string');
str = ['save ' name ' Te Tn Kp T sp pos'];
eval(str);
```

2.5. Segunda interfaz gráfica. Filtraje, Modelo Paramétrico y Reducción.

Con la implementación de la interfaz de excitación se conoce la PRBS y el comportamiento de las salidas correspondientes a la excitación sobre las articulaciones; por lo que se procederá al tratamiento de los datos, a la obtención del modelo paramétrico y a la reducción del modelo, para así completar el proceso de identificación.

2.5.1. Método para el desarrollo de la interfaz grafica.

En la primera interfaz gráfica se salvan los datos correspondientes a la PRBS y al comportamiento de la salida de las articulaciones, por tanto para la segunda interfaz gráfica es necesario cargar los datos provenientes de la interfaz de excitación para poder realizar el proceso de identificación de acuerdo a la señal de entrada y a la salida.

Es necesario cargar los datos salvados de la primera interfaz para filtrarla, luego de filtrarla se procede a la obtención del modelo paramétrico y de su porciento de seguimiento de la señal (FIT), así como la representación del sistema en lazo cerrado y también de la reducción del modelo a uno de tercer orden tipo1.

Primeramente se crea un control 'Push Button' que recibe el nombre de 'Cargar', el cuál carga los datos que corresponda con el nombre del archivo que el usuario salva. Luego de crearse el control de 'cargar' se crea otro control que se encarga de cargar los resultados salvados en la primera interfaz gráfica.

Cuando se cargan los resultados se comienza con el proceso de filtraje y el de obtener el modelo parametrizado, así también como su reducción.

Para mostrar el comportamiento de la señal de excitación y la respuesta del sistema ante esto, se utiliza el código de programación mostrado en los Anexos 1.

Después de haber cargado los datos y mostrar en el Axes el comportamiento correspondiente a eso resultados, se comienza con el filtraje y para ello es necesario introducir valores que contribuyan con dicha acción como son:

- Frecuencia de corte del filtro de Butterworth
- Tiempo de inicio
- Tiempo de finalización

Ya definidas las variables, se crea un control con el objetivo de ejecutar la acción de filtraje ver Anexo 1.

Luego de ejecutarse la acción de filtraje, se tiene la respuesta filtrada, entonces se comienza con la elección de los distintos tipos de estructuras paramétricas para obtener el modelo en discreto.

Después de definir el tipo de método que se desea aplicar y el orden del modelo a estimar (numerador, denominador), es necesario un 'Push Button' el cuál se encarga de compilar, para que se muestre el resultado en un Axes y a su vez calcule el porciento de seguimiento de la señal y muestra el rlocus del lazo abierto, y además nos calcula el modelo identificado, ver Anexo 2.

2.6. Conclusiones Parciales del Capítulo.

- Se ha podido ver hasta aquí las características de los requerimientos que se necesitan y además de los pasos a seguir para lograr el proceso de identificación. Se realiza un estudio de la herramienta del GUI, donde se describe desde su ejecución hasta las partes más importantes, así como la aplicación de algunos de sus controles.

- También se muestra la implementación del programa y la descripción en su diseño, que se realiza con el fin de crear la interfaz gráfica.

CAPÍTULO 3 RESULTADOS Y VALIDACIÓN.

La implementación de la interfaz gráfica para la identificación, propuesta en el capítulo anterior, es mostrada en este capítulo mediante su simulación en la plataforma de dos grados de libertad de SIMPRO. Para esto es necesario conocer como se emplea dicha interfaz para su correcto uso, para así obtener resultados satisfactorios: con curvas reales obtenidas experimentalmente se muestran los resultados prácticos obtenidos. Con lo anterior se valida el empleo de la interfaz gráfica para el proceso de identificación propuesto, en estructuras de dos grados de libertad como la del simulador de conducción mencionado.

3.1 Interfaz Grafica para la identificación

Como se abordó en el capítulo anterior, para identificar un sistema es conveniente realizarlo en lazo cerrado alrededor de diferentes posiciones, para ello se hace imprescindible la excitación de la planta con una señal PRBS, dicho proceso de excitación se logró implementar en una interfaz gráfica (Figura 3.1) que permite al usuario de forma interactiva conocer el comportamiento de la señal de entrada, así como el seguimiento de la respuesta ante esta excitación.

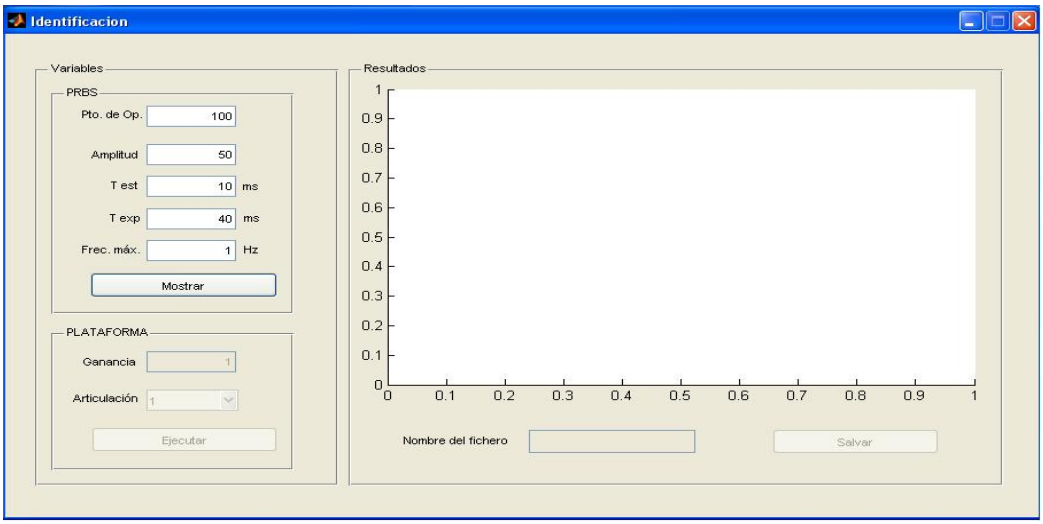


Figura 3.1. Interfaz de excitación.

Luego de llevar a cabo la excitación del sistema a través de la Figura 3.1, se procede a la identificación, pero para ello es necesario filtrar la señal y obtener su modelo paramétrico para luego proceder a la reducción del modelo (ver Figura 3.2).

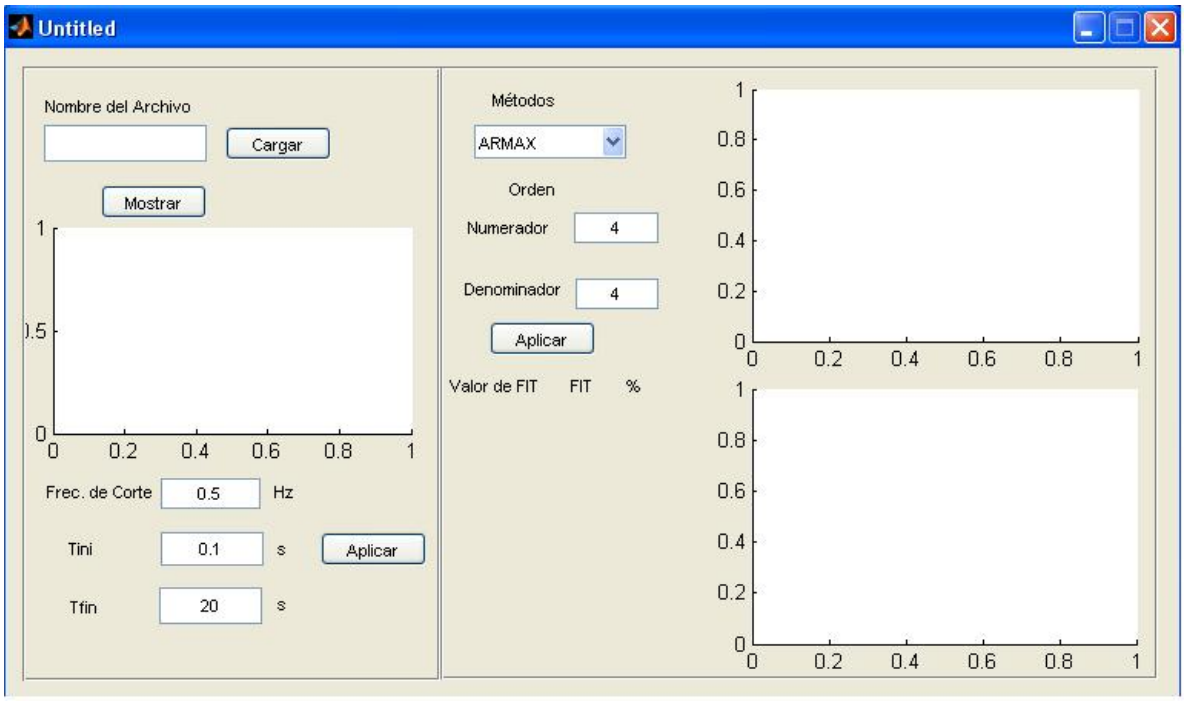


Figura 3.2. Filtraje, Modelo Paramétrico y Reducción.

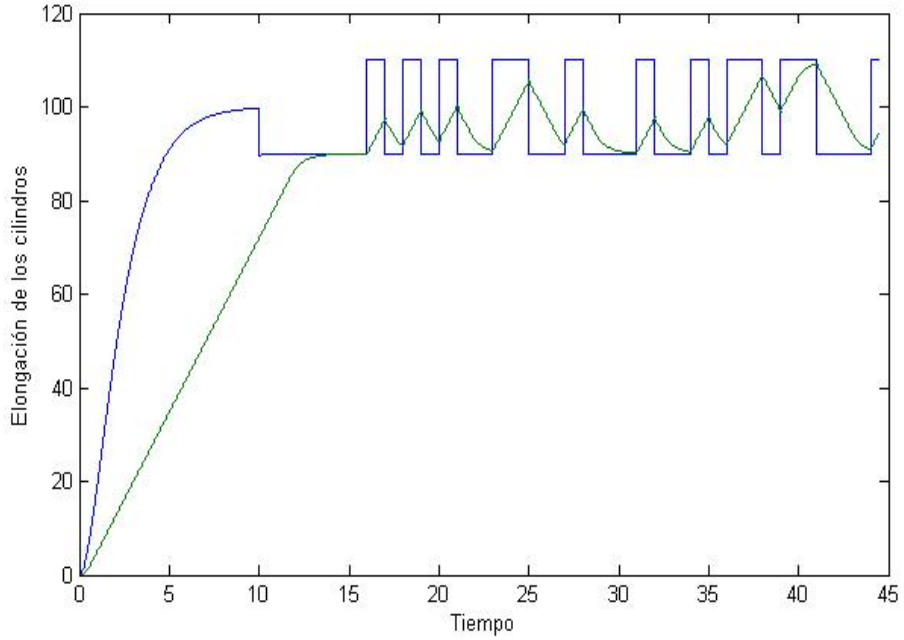
3.2. Simulación de la interfaz gráfica.

Luego de diseñada la interfaz se procede al uso de ella, para comprobar su correcto funcionamiento.

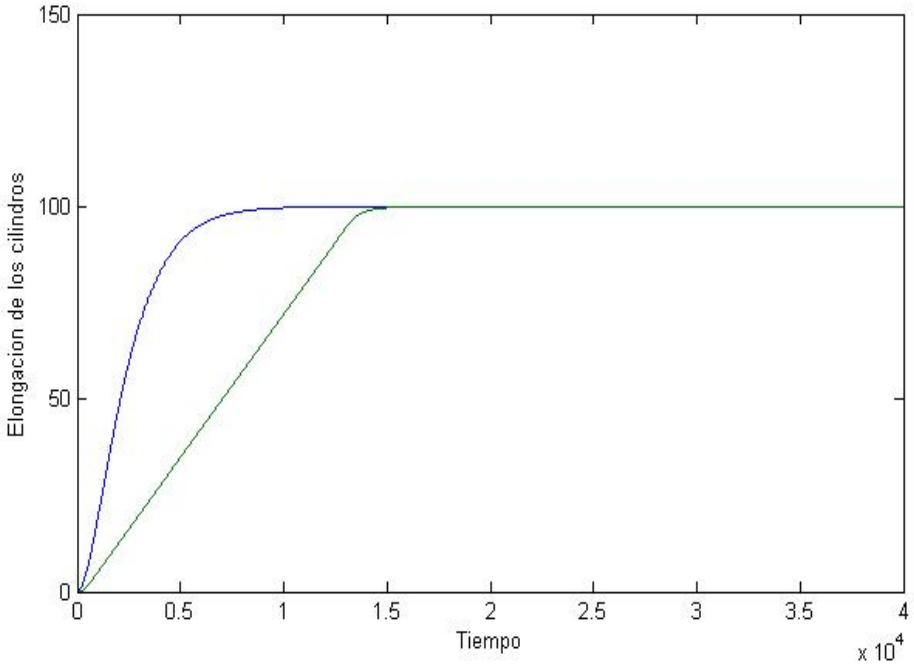
3.2.1. Interfaz gráfica 'Excitación del sistema'.

Primeramente se introducen los parámetros en la interfaz gráfica mostrada en la Figura 3.1, estos parámetros son cambiados en relación a como responda el sistema; en este caso se escoge los valores que por defecto aparecen en la interfaz, ya que como no se pudo probar en una plataforma real los resultados van a ser experimentalmente.

De acuerdo con los valores de los parámetros que en este caso se escogen, el resultado de la simulación fue el siguiente Figura 3.3 y Figura 3.4:



a)



b)

Figura 3.3. Excitación a la articulación 1 a) y excitación a la articulación 2 b).

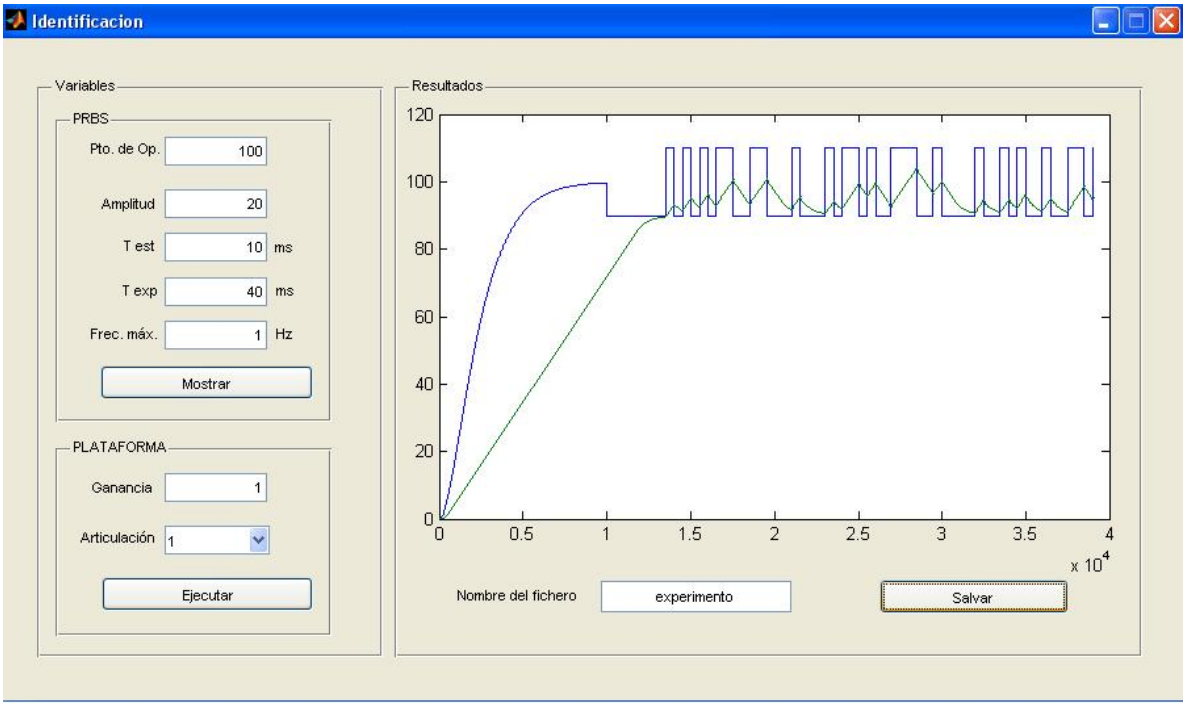


Figura 3.4. Interfaz gráfica de la excitación a la articulación 1.

Como se puede observar el fichero recibe el nombre de experimento para luego proceder a su salvamento. Si se desea comprobar que el archivo guardado contiene los datos necesarios para el siguiente proceso, se ejecuta experimento1.mat y se observa en el Workspace los datos que fueron guardados. Con el experimento anterior se logra comprobar el funcionamiento correcto de la primera interfaz gráfica 'Excitación del sistema'.

3.2.2. Interfaz gráfica 'Filtraje, Modelo Paramétrico y Reducción.

Después de completado el proceso de la primera interfaz gráfica, en donde se salvan los datos que se necesitan para trabajar con la segunda interfaz gráfica, se comienza con el análisis de implementación de esta segunda interfaz para así concluir con el proceso de identificación.

Primeramente se cargan los datos que fueron salvados en la primera interfaz y luego se muestran esos resultados, para después filtrar como se puede mostrar a continuación Figura 3.5:

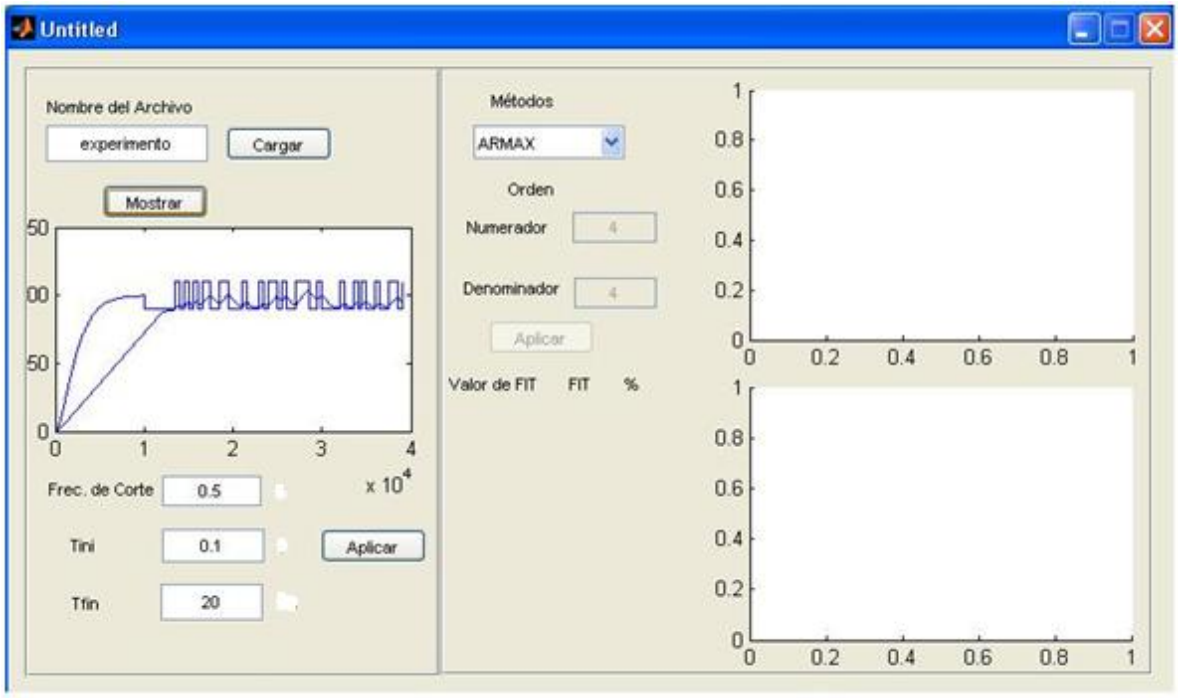


Figura 3.5. Muestra del resultado de la primera interfaz.

Ya mostrada la PRBS y la respuesta del sistema ante dicha señal, se procede al filtraje de la señal pues la señal proveniente de la primera interfaz puede presentar ruidos que me afecten con la respuesta final del proceso de identificación (Rubio, 2009). Para el filtraje de la señal se utilizó una frecuencia de corte del filtro de butterword fcb=0.5, un tiempo de inicio del experimento filtrado Tini=0.1 y un tiempo de finalizar el experimento Tfin=20, luego de definir estos parámetros que se muestran por defecto, a continuación se muestra el comportamiento de la señal filtrada Figura 3.6.

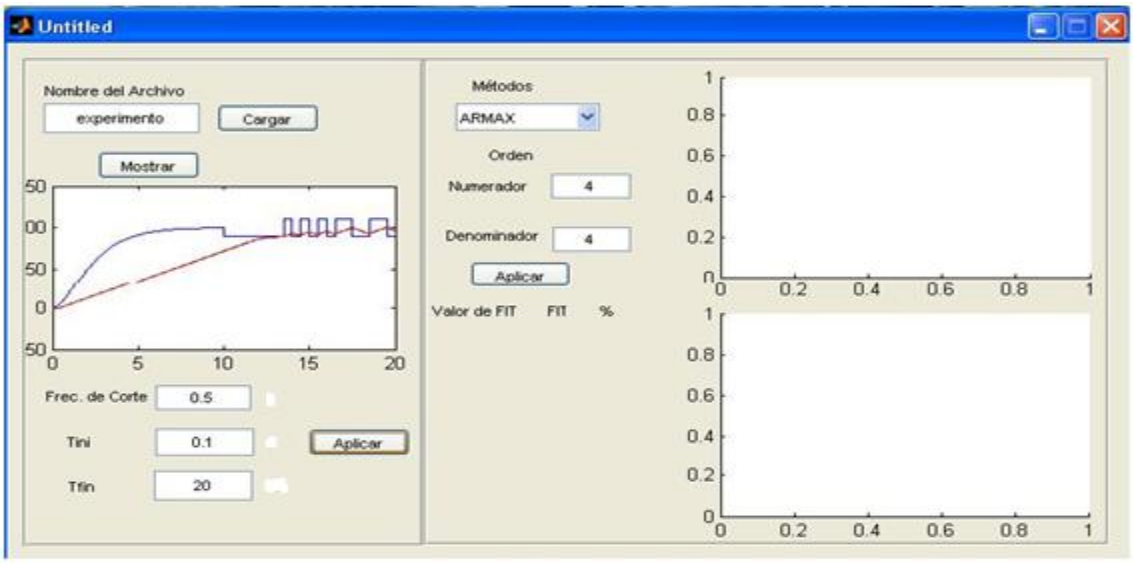


Figura 3.6. Comportamiento de la señal filtrada.

Una vez realizado el proceso de filtraje se pasa a la obtención del modelo en dependencia del método escogido, en este caso se escoge para la estimación un modelo ARMAX que da como resultado lo siguiente Figura 3.7. En esta figura se muestra el valor de FIT como también el rlocus del lazo cerrado.

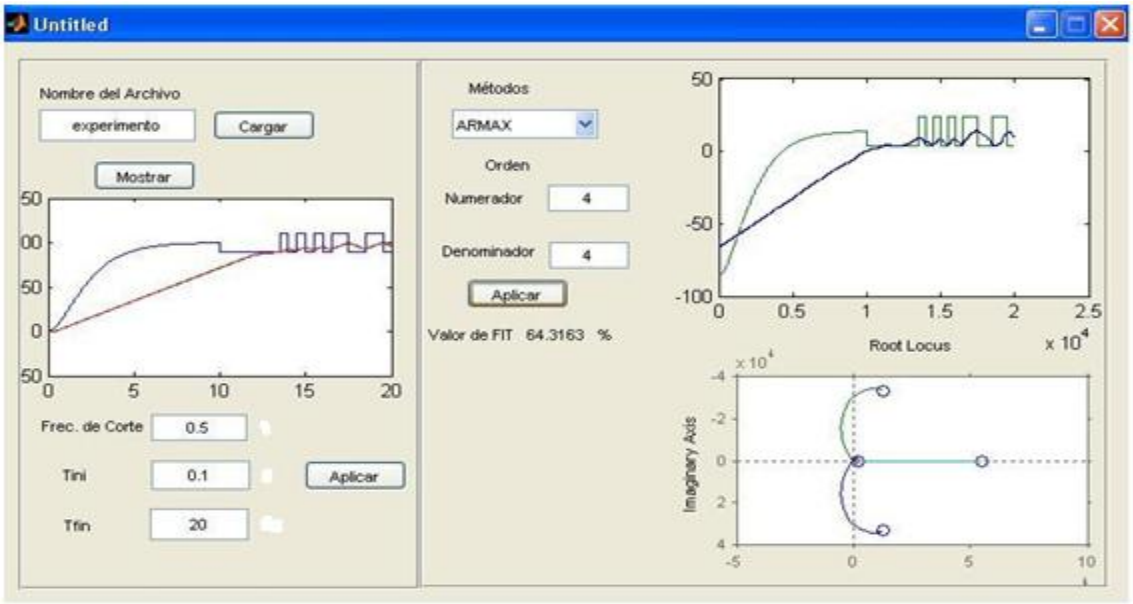


Figura 3.7. Interfaz con el método estimado, el valor de FIT y rlocus de lazo cerrado.

Luego de ejecutarse el Pushbutton ‘Aplicar’, en donde se muestra el resultado del comportamiento del modelo a partir de los parámetros estimados y también se muestra el rlocus del lazo abierto ver Figura 3.7, se obtiene el modelo identificado a partir de esta interfaz gráfica, el cuál es mostrado a continuación:

$$\text{sys} = \frac{0.548}{s^3+1.339s^2-0.04031s} \qquad \text{Ec.1.22}$$

Como se observa el modelo es de tercer orden de tipo 1 por lo que cumple con las exigencias impuestas, pues a través de las interfaces gráficas se logra conocer un modelo aproximado al de la planta, lográndose un correcto proceso de identificación.

3.3. Conclusiones parciales del capítulo.

- El software diseñado permite las estimaciones de un modelo electro neumático en tiempo real.
- El GUI es una herramienta que permite el manejo del proceso de identificación mucho más fácil, por lo que al usuario le resulta menos complicado en su manejo.
- Dicha interfaz muestra su utilización en cualquier simulador de dos grados de libertad y además siempre se obtiene un modelo de tercer orden de tipo 1 respondiendo a los objetivos de identificación.

CONCLUSIONES

Conclusiones

Con lo realizado en este trabajo podemos arribar a las siguientes conclusiones:

- El Matlab por sus características y su potencial en tiempo real, es el software necesario en la implementación de las interfaces gráficas para la identificación de los simuladores de conducción de dos grados de libertad.
- Se ha logrado diseñar una primera versión de las interfaces para la identificación, donde no es necesario dominar los diferentes pasos requeridos para la estimación de modelos electro-neumáticos.
- Se logra una simulación y obtención de un modelo electro-neumático con un porciento de seguimiento de 64.3%.

RECOMENDACIONES

- Profundizar en el estudio del manejo de la herramienta del GUI, enfocado a una mejora de los algoritmos de diseño e implementación de la interfaz.
- Extender el uso de dicho sistema para la identificación de plataformas paralelas de SIMPRO.
- Realizar pruebas experimentales en una plataforma real con el objetivo de obtener resultados reales.

REFERENCIAS BIBLIOGRÁFICAS

AGUADO, A. (2000) *Temas de Identificación y Control Adaptable,* La Habana, Cuba, ICIMAF. Instituto de Cibernética, Matemática y Física.

ARACIL, R., SALTARÉN, R., SABATER, J. M. & REINOSO, O. (2006) Robots Paralelos: Máquinas con un pasado para una Robótica del Futuro. *Revista Iberoamericana de Automática e Informática Industrial,* 3**,** 16-28.

COPLIEN, J. O., SCHMIDT, D. C., VLISSIDES, J. M., KERTH, N. L., MARTIN, R. C., RIEHLE, D. & BUSCHMANN, F. (1995) *Pattern languages of program design*.

ENTENZA, P. J. P. (2009) Diseño de una estructura paralela en ADAMS y su enlace con Simulink *Automatica y Sistemas Computacionales.*

HERNÁNDEZ, L., RUBIO, E., ARACIL, R., SALTARÉN, R. & ROSELL, K. (2004) Identificación y servo control robusto para cilindro actuador neumático. *Informática 2004.* La Habana, Cuba.

INSTRUMENTS, N. (2010) Características de Análisis y Generación de Reportes en LabVIEW 2010. 29-jul-2010

JANISZOWSKI, K. B. (2004) Adaptation, modelling of dynamic drives and controller design in servomechanism pneumatic systems. *IEE Proceedings - Control Theory and Applications,* 151**,** 234-245.

KARPENKO, M. & SEPEHRI, N. (2004) QFT Design of a PI Controller with Dynamic Pressure Feedback for Positioning a Pneumatic Actuator. IN IEEE (Ed. *IEEE American Control Conference*. Boston.

KOEKEBAKKER, S. H. (2001) *Model based control of a flight simulator motion system*. Univesidad de Toronto.

KRIVTS, I. L. & KREJNIN, G. V. (2006) *Pneumatic Actuating Systems for Automatic Equipment. Structure and Design,* Taylor & Francis Group.

KUMAR, S. (2010) Exploración de nuevas características de C++ y MFC en Visual Studio 2010. *MSDN Magazine.* Abril 2010.

LJUNG, L. (1999) *System Identification. Theory for the user,* Prentice Hall.

LÓPEZ, Y. R. (2008) Modelo Cinemático para Robot Paralelo de tres grados de libertad *Automática y Sistemas Computacionales.*

MACHADO, A. (2007) Controlador empotrado para plataforma neumática de simulador de conducción. *Departamento de Automática y Sistemas Computacionales.* Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

MACHADO, A., HERNÁNDEZ, L. & RUBIO, E. (2007) Controlador empotrado para plataforma neumática de simulador de conducción. *XIII Convención de Ingeniería Eléctrica.*

MATHWORKS (2007a) Control System Toolbox. Version 7.4 ed.

MATHWORKS (2007b) Real-Time Workshop. Version 7.4 ed.

MERLET, J. P. (2006) *Parallel Robots,* Springer.

MOORE, P. & PU, J. S. (1996) Pneumatic servo actuator technology. *IEE Seminar Digests,* 1996**,** 3.

MORENO, R. (2000) Plataforma para Simuladores. *SIMPRO.* Ciudad de la Habana, Cuba, Instituto Superior Politécnico "José Antonio Echevarría".

PLATTENBURG, D. H. (2005) Pneumatic actuators: a comparison of energy-to-mass ratio's. *ICORR 2005. 9th International Conference on Rehabilitation Robotics.*

PRATTICHIZZO, D. B., A.**,** (Ed.) (1998) *Dynamic analysis of mobility and graspability of general manipulation systems.*

RUBIO, A. E., HERNANDEZ, L., ARACIL, R., SALTARÉN, R. & GUERRA, J. A. (2009) Implementation of Decoupled Model-Based Controller in a 2-DOF Pneumatic Platform used in Low-Cost Driving Simulators. *IEEE Electronics, Robotics and Automotive Mechanics Conference, CERMA '09.* Morelos, Mexico.

RUBIO, E. (2009) Modelación, identificación y control de actuadores electro-neumáticos para aplicaciones industriales. *Departamento de Automática y Sistemas Computacionales.* Santa Clara, Universidad Central "Marta Abreu" de Las Villas.

RUBIO, E., HERNÁNDEZ, L., ARACIL, R. & SALTARÉN, R. (2007a) *Modelado, identificación y control de actuador lineal electro-neumático con válvula subdimensionada,* Ciudad de la Habana, Cuba, ISPJAE.

RUBIO, E., HERNÁNDEZ, L., ARACIL, R., SALTARÉN, R. & MORENO, R. (2007b) Modelado, identificación y control de actuadores lineales electro-neumáticos. Aplicación en plataforma de dos grados de libertad. *Revista Iberoamericana de Automática e Informática Industrial,* 4**,** 58 - 70.

RUBIO, E., HERNÁNDEZ, L., ARACIL, R., SALTARÉN, R. & ROSELL, K. (2005) Identificación y servo control robusto para sistema electro neumático. *XII Simposio de Ingeniería Eléctrica.* Santa Clara, Cuba.

RUBIO, E., HERNÁNDEZ, L., MACHADO, A. & MORENO, R. (2007c) Identificación y control de actuadores lineales electro neumáticos en simulador de conducción. *XIII Convención de Ingeniería Eléctrica.*

RUBIO, E., REINOSO, O. & SALTAREN, R. (2001) Identificación experimental de un sistema electro-neumático. *METANICA 2001*. La Habana, Cuba.

VARSEVELD, R. V. & BONE, G. (1997) Accurate position control of a pneumatic actuator using on/off solenoid valves. *IEEE/ASME Transactions on Mechatronics,* 2.

VELAZCO, S. E. D. (2007) Modelo Cinemático y Dinámico para Plataforma de dos grados de libertad *Departamento de Automática y Sistemas Computacionales*

YAMADA, Y., TANAKA, K. & UCHIKADO, S. (2000) Adaptive pole-placement control with multi-rate type neural network for pneumatic servo system. *IEEE Proceedings of the International Conference on Control Applications.*

ANEXO 1

Secuencia de código de la segunda interfaz gráfica

- Código para mostrar el comportamiento de la señal de excitación y la respuesta del sistema, luego de cargar los archivos que se desea:

Pushbutton_mostrar:

```
%Colocar Imagen de fondo
figura1 = imread('a.jpg'); %Leer imagen
axes(handles.figura1); %Carga la imagen en background
imshow(figura1); %Presenta la imagen
handles.output = hObject;
guidata(hObject, handles);
plot(sp);
hold on
plot(pos);

hold off
```

- Ejecutar acción de filtraje:

```
Ni = Tini/T;
Nf = (Tini+Tfin)/T;
Nt = Nf-Ni;
Nm = round(Nt/2);
sp = sp(Ni:Nf)-PO;
pos = dtrend(pos(Ni:Nf));
[B,A] = butter(2,fcb*T/2);
posf = filter(B,A,pos);
id = [posf sp];
id = dtrend(id);% Remover o no la media en id
tim = (0:Nt)'*T;
save test posf tim;
figura1 = imread('a.jpg'); %Leer imagen
axes(handles.figura1); %Carga la imagen en background
imshow(figura1); %Presenta la imagen
handles.output = hObject;
guidata(hObject, handles);
plot(tim,[sp pos posf]);
figure
plot(tim,[sp pos posf]);
```

ANEXO 2

- Código de implementación de los métodos y obtención del FIT:

Push Button:

```
idd = id;
idde = idd;
iddv = idd;
n = 4;
metodo = 'ARMAX';  %  OE, BJ, ARX, ARMAX
idd = id;
idde = idd;%idd([1:Nm],:);
iddv = idd;%idd([Nm:Nt],:);

switch upper(metodo)

case 'OE'
    mod = oe(idde,[n n 1]);
case 'BJ'
    mod = bj(idde,[n n n n 1]);
case 'ARX'
    mod = arx(idde,[n n 1]);
case 'ARMAX'
    mod = armax(idde,[n n n 1]);
end

mod.Ts = T;
figure
resid(mod,iddv)
figure
compare( iddv,mod )
[YH,FIT] = compare( iddv,mod );
mod.Ts = T;
present(mod)
resid(mod,iddv)
figure
compare( iddv,mod )
[YH,FIT] = compare( iddv,mod );
set(handles.fit,'String',FIT)
err = [FIT mod.EstimationInfo.LossFcn];
idmod = mod;
%Colocar Imagen de fondo
figura2 = imread('a.jpg'); %Leer imagen
axes(handles.figura2); %Carga la imagen en background
%axis off;
imshow(figura2); %Presenta la imagen
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
%Aqui pongo lo que tengo que graficar
plot(iddv)
save modelid err idmod;
```

- Código de implementación para convertir a continuo y mostrar el rlocus del lazo abierto:

Pushbutton:

```
load experimento
load modelid
%str = ['load ' datafile ';'];
%eval(str);
id = idmod;
switch upper(metodo)
 case {'OE','BJ'}
      ndcl = id.B;
      ddcl = id.F;
 case {'ARX','ARMAX'}
      ndcl = id.B;
      ddcl = id.A;
 end
SYSD = TF(ndcl,ddcl,0.001);

SYSC = D2C(SYSD,'tustin');

[ncl,dcl] = tfdata(SYSC,'v');
nol = ncl/Kp;
dol = dcl-ncl;
nol = nol/dol(1);
dol = dol/dol(1);
sys = tf(nol,dol);
modol = sys
num = nol;
den = dol;
zpk(modol)
roots(den)
figura3 = imread('a.jpg'); %Leer imagen
axes(handles.figura3); %Carga la imagen en background
%axis off;
imshow(figura3); %Presenta la imagen
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
%Aqui pongo lo que tengo que graficar
rlocus(sys)
save modelo1 num den modol Kp;
```

- Código de implementación para reducir el modelo a uno de tercer orden de tipo 1:

```
load modelo1;
[modb,g] = balreal(modol);
 if length(g)>3,
    modr1 = modred(modb,4:length(g),'MatchDC');
    modr2 = modred(modb,4:length(g),'Truncate');
 else
```

```
   modr1 = modb;
   modr2 = modb;
end
 [Z,P,K] = zpkdata(modr2,'v')
 K*Z(1)*Z(2)*Z(3)/(P(1)*P(2)*P(3))
 abs(P(2))/(2*pi)
  if real(P(1)) == P(1)
     modr3 = zpk([],[P(2:3);0],K*Z(1)*Z(2)*Z(3)/P(1));
  else
     modr3 = zpk([],[P(1:2);0],K*Z(1)*Z(2)*Z(3)/P(3));
  end
zpk(modr1)
zpk(modr2)
zpk(modr3)
[n,d] = tfdata(modr3,'v');
sys = tf(n,d)
km  = real(n(4)/d(3))
a1m = d(2)
a0m = d(3)
figure
bode(modol,modr1,modr2,modr3,sys), grid on;
save modelol num den modol modr1 modr2 modr3 km a0m a1m Kp;
```