Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Interfaz para la identificación y ajuste de los controladores de un simulador de conducción.

Autor: Ariel Adrian Valdivia Pacheco

Tutor: Dr.C. Angel Ernesto Rubio Rodríguez

Santa Clara

2012

"Año 54 de la Revolución"

Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Interfaz para la identificación y ajuste de los controladores de un simulador de conducción.

Autor: Ariel Adrian Valdivia Pacheco
email: avpacheco@uclv.edu.cu

Tutor: Dr.C. Angel Ernesto Rubio Rodríguez Prof. Titular
Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV
email: rubio@uclv.edu.cu

Santa Clara

2012

"Año 54 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.

Ariel Adrian Valdivia Pacheco — Fecha
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.

Ariel Adrian Valdivia Pacheco — Fecha
Autor

Boris Luis Martínez Jiménez, Dr.C — Fecha
Jefe del Departamento

Responsable ICT o J' de Carrera, (Dr.C., M.Sc. o Ing.) — Fecha
Responsable de Información Científico-Técnica

PENSAMIENTO

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica."

Aristóteles

DEDICATORIA

A mi abuela, **Maritza**,
la persona más importante en mi vida.

Por tu cariño y apoyo incondicional,
mostrándome siempre el camino a seguir,
por haber hecho de mí, el hombre que soy,
y sobre todo por estar siempre a mi lado.

Este sueño hecho realidad,
que ha conllevado mucho sacrificio de tu parte mami,
Nunca hubiera sido posible sin tu ayuda.
TE QUIERO MUCHO

AGRADECIMIENTOS

A mis abuelas Maritza e Isabel, por quererme tanto y confiar en mí.

A Baño, que para mí es como mi abuelo, por inculcarme siempre los deseos de estudiar.

A mi mamá y mi papá, por haberme regalado lo más preciado, la vida.

A mi novia Yisel, por su amor y comprensión.

A mi tutor Ernesto Rubio, por su disposición y dedicación en la realización de este Trabajo de Diploma.

A toda mi familia, por el apoyo brindado.

A todos los profesores, por los conocimientos y los valores inculcados.

A mis amigos, por todos estos años que vivimos juntos y la gran hermandad que nos une.

En fin, a todos los que de una forma u otra me han ayudado y han hecho posible que haya llegado hasta aquí.

Santa Clara, Cuba, 2012

RESUMEN

El proceso de identificación y control de los sistemas en general es complejo. En los simuladores de conducción basados en actuadores electro-neumáticos constituye un problema especialmente complejo. El presente trabajo de diploma parte de la necesidad de diseñar una interfaz gráfica capaz de realizar el proceso de identificación y control, haciendo de una forma más sencilla la estimación de los modelos y síntesis de los controladores para los simuladores de conducción de dos grados de libertad desarrollados por el Centro de Investigación y Desarrollo de Simuladores (SIMPRO).

El objetivo general es: desarrollar una interfaz amigable para la realización semiautomática del proceso de identificación y el ajuste de los controladores de los actuadores electro-neumáticos del simulador de conducción de SIMPRO.

La interfaz se implementó en *MATLAB* 7.8, utilizando la herramienta de Interfaz Gráfica de Usuarios (GUI, *Graphical User Interface*), que permite crear un ambiente gráfico, y aprovechando sus potencialidades para ejecutar tareas en tiempo real. La interfaz, en las pruebas reales realizadas al simulador de SIMPRO muestra excelentes resultados.

TABLA DE CONTENIDO

Página

ÍNDICE DE FIGURAS

Figura Página

INTRODUCCIÓN

Un simulador es un aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema, con ellos se simulan sensaciones físicas (aceleración, percepción del entorno, velocidad) y situaciones de emergencia sin poner en peligro el sistema real ni las personas.

Estos han tenido gran auge en el mundo entero, tanto es así, que podemos encontrar simuladores de vuelo, de conducción, de carreras, de trenes, entre otros. Ellos permiten el entrenamiento del personal y minimizan el uso de vehículos reales, con el consiguiente ahorro de recursos y seguridad para el personal.

Entre los más desarrollados se encuentran los simuladores de conducción, que además de ser usados para la capacitación y entrenamiento de potenciales conductores de vehículos, tienen una aplicación creciente en los juegos, aumentando su aceptación en la medida en que logran recrear con mayor precisión la inmersión en un mundo virtual. Los más complejos recrean incluso las sensaciones de movimiento con plataformas móviles de varios grados de libertad (GDL). Es importante destacar que evitan posibles accidentes de los principiantes en condiciones de peligrosidad, pues el ordenador permite reproducir situaciones extremas de conducción en un ambiente virtual, seguro y totalmente controlado (Aracil, 2006).

La gran mayoría de los simuladores comerciales son altamente costosos por el conocimiento que entrañan y el uso de actuadores hidráulicos o eléctricos. Son una solución económica para las plataformas de simuladores la utilización de actuadores neumáticos lineales, que

tienen como ventajas la limpieza, altas razones de carga contra peso y carga contra volumen, altas velocidades y grandes fuerzas, todo esto logrado a través de un mecanismo operacional simple (Quintana, 2000; Murthy, 2009).

En Cuba, el Centro de Investigación y Desarrollo de Simuladores (SIMPRO), brinda solución a las necesidades del país, al producir simuladores de conducción ampliamente utilizados en entrenamiento de personal de la producción y los servicios, así como en juegos virtuales (Quintana, 2000; Entenza, 2009) pero las prestaciones actuales de los mismo son limitadas porque no cuentan con una estrategia de control adecuada (Rubio, 2009).

Los simuladores desarrollados por SIMPRO están basados en estructuras mecánicas paralelas accionadas por cilindros electro-neumáticos. Su ventaja es la neumática que abarata costos y simplifica el mantenimiento, pero su utilización trae consigo serios problemas de control por las no linealidades inherentes a la neumática y las estructuras paralelas (Rubio, 2009).

Al Grupo de Automatización, Robótica y Percepción (GARP) del Departamento de Automática de la UCLV, se le propone por parte de SIMPRO resolver el problema de control de estas estructuras, para lo cual se ha desarrollado 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 dicho modelo (Rubio, 2009; Hernández, 2004).

Actualmente el proceso de identificación y ajuste de los controladores debe hacerlo un especialista, pues no existe una interfaz que permita hacer cómodamente este proceso. De modo que se hace necesario diseñar una interfaz gráfica amigable que permita realizar el proceso de identificación y ajuste de los controladores para poder generalizar su uso a nivel nacional.

En consecuencia con esta problemática, se establecen como objetivos de este trabajo, los siguientes:

Objetivo general:

⋆ Desarrollar una interfaz amigable para la realización semiautomática del proceso de identificación y el ajuste de los controladores de los actuadores electro-neumáticos del simulador de conducción de SIMPRO.

Objetivos específicos:

⋆ Analizar críticamente las alternativas existentes para el desarrollo de la interfaz necesaria.

⋆ Implementar una interfaz amigable que cumpla con los requerimientos necesarios.

⋆ Validar la interfaz desarrollando todo el proceso en un simulador real.

Los resultados del desarrollo de esta interfaz son una aplicación práctica y teórica de gran trascendencia en los simuladores de conducción basados en actuadores neumáticos y un posible impacto en la colaboración con la empresa SIMPRO en una producción masiva de simuladores de conducción.

Organización del informe:

El siguiente informe, posterior a la introducción, cuenta con tres capítulos, conclusiones y recomendaciones, referencias bibliográficas y anexos. El **primer capítulo** trata el marco teórico referencial donde se realiza un análisis de los simuladores desarrollados en el mundo y los actuadores que emplean, la importancia de los simuladores de conducción y las características del simulador de SIMPRO, la necesidad de una interfaz que permita realizar una identificación y sintetizar un control de forma más sencilla. Se plantean los requerimientos técnicos así como las ventajas y desventajas de las alternativas existentes para el desarrollo de la interfaz.

En el **segundo capítulo** se describe el diseño de la interfaz, que tiene como característica: permitir una buena identificación del sistema; obtención y validación del modelo; así como la síntesis y puesta a punto del controlador.

En el **tercer capítulo** se presenta la aplicación y validación de la interfaz en un simulador real; y se dan a conocer los resultados obtenidos a partir de los experimentos realizados con la interfaz en tiempo real.

Finalmente, se exponen las **conclusiones** y **recomendaciones** a las que se arribó tras el desarrollo de este trabajo, así como la **bibliografía** referenciada a lo largo del informe y los **anexos** relevantes.

CAPÍTULO 1
MARCO TEÓRICO REFERENCIAL

La interfaz se aplicará en el simulador de conducción de SIMPRO que está basado en una estructura que puede considerarse un *robot* paralelo. Por ello es necesario realizar una breve reseña histórica acerca de la evolución de este tipo de *robots* y su aplicación en los simuladores de conducción lo cual ha tomado gran auge en el mundo.

1.1. Importancia de los simuladores de conducción.

Los simuladores de conducción tienen una aplicación creciente en juegos y entretenimientos. Contribuyen al aprendizaje de conductores pues los capacita para manejar sus coches, evitando así, posibles accidentes de los principiantes en condiciones de peligrosidad. Esto se debe a que el ordenador reproduce situaciones extremas de conducción en un ambiente virtual, seguro y totalmente controlado. Los simuladores permiten que se minimice de manera real la utilización de los vehículos (Aracil, 2006) conllevando al ahorro de recursos financieros pues no se producen gastos por combustible ni se daña el medio ambiente, además, la experiencia alcanzada por los conductores contribuye a que no ocurran accidentes en las vías, una de las principales causas de pérdidas de vidas humanas en la actualidad.

1.2. Simuladores desarrollados en el mundo y actuadores que emplean.

La base de los simuladores de conducción son los *robots* paralelos. El primero 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* (Merlet, 2006; López, 2010).

En 1947 *V.E. Gough* ideó un *robot* paralelo con seis actuadores lineales mostrado en la figura 1–1, formando una estructura de octaedro (Aracil, 2006; Merlet, 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, 2010).

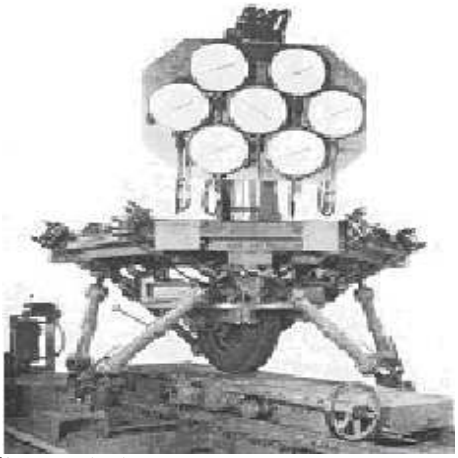


Figura 1–1: *Robot* paralelo de Gough.

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 figura 1–2 (Velazco, 2007).

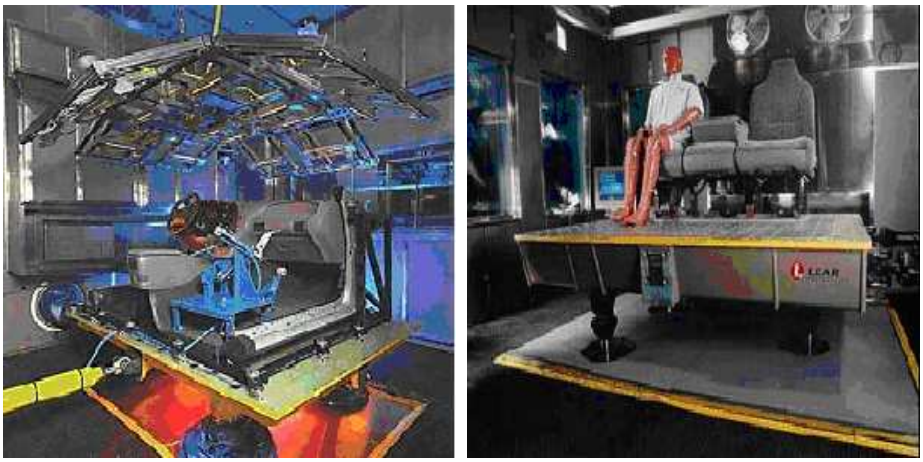


Figura 1–2: Varios tipos de mesas multiejes para la simulación de vehículos.

Los actuadores están clasificados en tres grandes grupos: neumáticos, hidráulicos y eléctricos. Los actuadores neumáticos son clasificados en cilindros neumáticos de simple y doble

efecto y motores neumáticos de aletas rotativas y pistones axiales, mientras que los actuadores hidráulicos en cilindro y motor hidráulico y los actuadores eléctricos en motores lineales o rotacionales.

Los actuadores eléctricos tradicionalmente se usan para el posicionamiento continuo de las cargas porque son relativamente fáciles de controlar, pero tienen una baja relación potencia peso y son mucho más caros especialmente los motores lineales. Los hidráulicos se usan cuando hay que manipular cargas grandes dada su alta relación potencia peso, pero son un poco más difíciles de controlar y complejos a la hora del mantenimiento. Los neumáticos son para cargas relativamente grandes dada su alta relación potencia peso, son fáciles de mantener pero difíciles de controlar cuando se requiere posicionamiento continuo.

1.3. Simulador de SIMPRO.

La empresa SIMPRO produce simuladores de conducción basados en Realidad Virtual para el entrenamiento de personal. Los mismos están constituidos por una cabina equipada con mandos reales y un monitor, que simulan el estado del vehículo en un entorno virtual y pueden llevarse a cabo simulaciones tanto para vehículos pesados como para ligeros (Carballo, 2011).

El simulador de SIMPRO en el que se va a trabajar, es un *robot* paralelo de dos grados de libertad (Figura 1–3) es una plataforma que presenta un diseño formado por una estructura paralela, con cinco uniones cardánicas y dos articulaciones constituidas por cilindros neumáticos, que son los encargados de darle al conductor las sensaciones de ladeo y cabeceo (Rubio, 2009).

Estos simuladores tienen como condición crítica de movimiento a experimentar por el vehículo simulado, 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/s^2 que para los cilindros representa aceleraciones lineales de hasta 1000 mm/s^2.

En la Tabla 1–1 se pueden encontrar los datos más importantes de la plataforma del simulador (Quintana, 2000). El origen de coordenadas para las medidas de longitud y

Figura 1–3: Simulador de SIMPRO de dos grados de libertad.

ubicación del centro de masa se establece en el pivote central. 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: Datos del simulador de SIMPRO.

Parámetros	Valor
Masa total de la cabina	215 Kg
Posición del centro de masa X	100 mm
Posición del centro de masa Y	60 mm
Posición del centro de masa Z	480 mm
Distancia del origen de cada cilindro	560 mm
Elongación de los cilindros	$\pm$150 mm
Ángulo de ladeo y cabeceo	$\pm$0,26 rad

1.3.1. Características y problemática.

En la robótica paralela una solución económica es la utilización de actuadores electro-neumáticos lineales, que tienen como ventajas la limpieza, altas razones de carga contra peso y carga contra volumen, bajo costo, respuesta rápida y grandes fuerzas, todo lo anterior logrado a través de un mecanismo operacional simple. Esto permite el desarrollo de manipuladores compactos, ligeros y rentables, que pueden ser utilizados en una gran variedad de aplicaciones como los simuladores de conducción (Quintana, 2000; Carducci, 2004; Chen, 2004; Aracil, 2003) quedando económicamente viables como producto comercial.

El posicionamiento continuo de las cargas, con actuadores electro-neumáticos, ha resultado ser un problema de control complejo. Esto se debe fundamentalmente a que la dinámica de

los actuadores electro-neumáticos es altamente no lineal debido a la compresibilidad del aire, el comportamiento no lineal del flujo de aire a través de las válvulas y la existencia de elevadas fuerzas de fricción estáticas y dinámicas entre el cilindro y el pistón (Pearce, 2005).

Lograr un buen desempeño de los actuadores electro-neumáticos en aplicaciones donde se requiere el control continuo de su posición a todo lo largo del cilindro, es una problemática. De ahí que establecer un método para la síntesis de un controlador sencillo que garantice adecuadas prestaciones sin la necesidad de sensores adicionales, resulta algo muy conveniente para aplicaciones prácticas. Si, por demás, ese método es generalizable a sistemas de varios grados de libertad, su utilidad resulta aún mayor (Rubio, 2009).

1.3.2. Solución de control.

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 dadas las no linealidades intrínsecas de la tecnología neumática.

La principal tarea de GARP es el diseño e implementación de la estrategia y el sistema de control de los actuadores electro-neumáticos que accionan los simuladores (Rubio, 2009). Para ello se ha desarrollado una propuesta que parte de la identificación experimental del sistema, con el objetivo de estimar el modelo dinámico de la planta, para luego poder sintetizar un controlador robusto basado en dicho modelo (Rubio, 2009; Hernández, 2004).

1.3.3. Excitación de la planta en lazo cerrado.

La dinámica del sistema es claramente dependiente de la posición de la carga y el sistema a identificar es de tercer orden tipo uno, por lo que resulta conveniente identificarlo en lazo cerrado alrededor de diferentes posiciones utilizando un controlador proporcional, ya que en lazo abierto la respuesta del sistema tiende al infinito.

1.4. Definición de requerimientos.

Actualmente el proceso de identificación y ajuste de los controladores del simulador de SIMPRO debe hacerlo un especialista. Los resultados obtenidos por GARP pretenden ser generalizados por SIMPRO y para ello se requiere transferir la tecnología de la forma más sencilla. Se hace necesario diseñar una interfaz gráfica amigable y relativamente sencilla que permita realizar el proceso de identificación y ajuste de los controladores para poder generalizar su uso a nivel nacional. La interfaz tiene que tener un ambiente comprensible para cualquier usuario que con un mínimo de experiencia sobre el manejo de las plataformas pueda identificar su función de transferencia y sintetizar el controlador de la misma.

En el proceso de identificación y control de los sistemas no puede faltar el diseño de la señal de excitación así como su procesamiento previniendo datos erróneos, la comunicación de la interfaz con la plataforma, los métodos de identificación y validación a utilizar en una correcta estimación del modelo con el cual se sintetiza el control para luego probar su desempeño en la plantar real.

1.4.1. Generación de PRBS.

La PRBS (*Pseudo Random Binary Sequence*) es una señal binaria *pseudo* aleatoria, que tiene un espectro de frecuencias controlado por el usuario (Ljung, 1999). Esta señal se aplica a la entrada del sistema hasta provocar una excitación persistente de forma que los datos recogidos proporcionen toda la información posible sobre la dinámica de la planta.

Para desarrollar el proceso de identificación del sistema es necesario registrar la señal obtenida a la salida después de ser excitado por la PRBS. Con ambas señales, aplicando los métodos de estimación de parámetros, se determina el modelo dinámico discreto que las correlacione adecuadamente (Rubio, 2009).

1.4.2. Comunicación con tarjeta de adquisición de datos.

Se utiliza una tarjeta de adquisición de datos *HUMUSOFT MF614* la cual brinda una gran potencia de cálculo y admite el control de ambas articulaciones con un período de

muestreo de 1 ms. Mediante ella se reciben las elongaciones de los actuadores de ladeo y cabeceo de la plataforma.

1.4.3. Pre-procesamiento de la señal medida.

El proceso de identificación se puede ver afectado por presencia de perturbaciones de alta frecuencia, que son las que están por encima de las frecuencias de interés en la respuesta del sistema, también por datos erróneos, producidos por fallos en el *hardware* o *software* utilizados en el experimento de recogidas de muestras así como perturbaciones de baja frecuencia y todo esto puede causar conclusiones equivocadas sobre el comportamiento del sistema en cuestión.

Antes de proceder a realizar la estimación del modelo 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 (Ljung, 1999).

1.4.4. 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). Generalmente estos modelos permiten describir el comportamiento de cualquier sistema lineal. La dificultad radica en la elección del tipo de modelo (estructura, número de parámetros, etc.) que se ajuste satisfactoriamente a los datos de entrada-salida obtenidos experimentalmente (Carballo, 2011).

Los métodos implementados con los cuales se puede lograr una correcta identificación del sistema son:

⋆ ARX (*Auto Regressive and Exogenous Variable*). Se resuelve directamente por el algoritmo de mínimos cuadrados.

⋆ ARMAX (*Auto Regressive Moving Average and Exogenous Variable*). Se resuelve minimizando el error de predicción de forma iterativa, aplicando el algoritmo de mínimos cuadrados extendido.

⋆ OE (*Output Error*). Se resuelve con un algoritmo similar al ARMAX modificando el cálculo del error de predicción y el gradiente.

⋆ BJ (*Box-Jenkins*). Se resuelve con un algoritmo similar al ARMAX modificando el cálculo del error de predicción y el gradiente.

La obtención de un modelo de este tipo se realiza en dos pasos principalmente: primeramente tener una idea del sistema a identificar para hacer una correcta elección de la estructura del modelo y luego obtener los parámetros del modelo que ajustan la respuesta del mismo a los datos de entrada y salida obtenidos experimentalmente.

1.4.5. 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 mencionan algunos criterios típicos a la hora de descartar o elegir unos modelos respecto a otros: validación en base a la simulación del modelo, comprobación de parámetros estadísticos, chequeo de la ubicación de las raíces y reducción del modelo, simulación y análisis de residuos.

1.4.6. Síntesis del controlador.

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 (Rubio, 2009).

Se requiere que la interfaz permita comprobar el desempeño del controlador sintetizado sobre la planta real.

1.5. Alternativas existentes 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 Studio 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 (Carballo, 2011). Para la elección del *software* donde implementar la interfaz es necesario realizar una comparación entre *Visual Studio C++*, *Labview* y *MATLAB* para analizar ventajas y desventajas y así ver que cumplan con los requerimientos necesarios.

El *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*. Incluye las MFC (*Microsoft Foundation Classes*), equivalentes a las OWL (*Object Window Library*) de Borland. Estas clases facilitan la programación, sin tener que utilizar directamente el API de *Windows*, ya que agrupan las clases de *C++*. MFC es una jerarquía de clases *C++*, 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, 1995). 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 sobre *Windows* (Kumar, 2010).

Mientras el *Labview* es una herramienta gráfica para pruebas, control y diseño mediante la programación. Usa el lenguaje *G*, donde *G* representa un lenguaje gráfico. Este programa fue creado por *Nationals Instruments* (1976) para funcionar sobre máquinas MAC. 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 (Instrument, 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.

A pesar de tener grandes prestaciones no tiene bibliotecas para el proceso de obtención y validación del modelo. También al igual que *C++* genera un ejecutable pero necesita un núcleo mínimo del *software* base (Instrument, 2010).

Por otra parte se tiene el *MATLAB* que 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 frecuentes de *MATLAB* son: matemática y computación,

desarrollo de algoritmos, modelado y simulación, análisis de datos, gráficas científicas e ingenieriles, exploración y visualización, desarrollo de aplicaciones, incluyendo el diseño de interfaces gráficas de usuario, entre otras (Carballo, 2011).

El *MATLAB* presenta una familia de soluciones a aplicaciones específicas de acoplamiento rápido llamadas paquetes de herramientas (*toolboxes*). Los *toolboxes* son colecciones muy comprensibles de funciones o archivos (*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, el de señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc. (Carballo, 2011).

El *toolbox* de identificación de sistemas (*Systems Identification*) de la Figura 1–4, contiene funciones para la estimación de modelos de sistemas dinámicos tiene implementado métodos de estimación de modelos paramétricos y no paramétricos, brinda soporte a muchas clases de modelos lineales y no lineales, provee un ambiente interactivo para análisis de datos, estimación y validación de modelos, facilita la estimación de modelos continuos y discretos, usando datos en el dominio del tiempo.

El *toolbox* para *SIMULINK/Real Time Windows Tarjet* que permite la comunicación de la computadora en tiempo real a través de una tarjeta de adquisición de datos.

Entre las principales características del *MATLAB* 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.

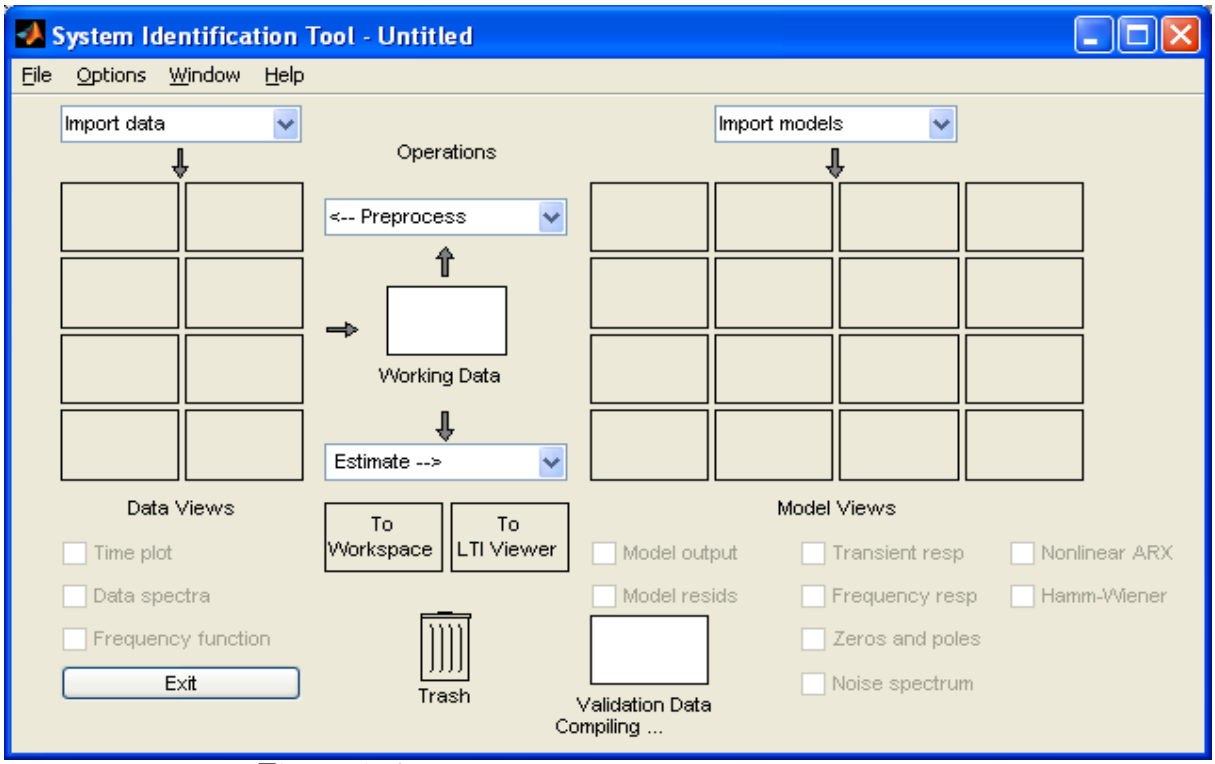


Figura 1–4: Herramienta de identificación de sistemas.

⋆ Aplicaciones especializadas para algunas ramas de ciencias e ingeniería (*toolboxes*).

El *MATLAB* cuenta entre sus ventajas con un conjunto de herramientas para el desarrollo de interfaces gráficas, las mismas permiten mejorar el ambiente en el que se ejecuta una determinada aplicación programada en *MATLAB*.

La herramienta matemática más completa debido a su amplia aplicación, es el *MATLAB*, aunque presenta como desventaja el hecho de necesitar plataforma de *software* para llevar a cabo la simulación (Carballo, 2011) es el usado en el desarrollo de la interfaz de identificación y control de los simuladores de conducción de dos grados de libertad de SIMPRO.

1.6. Consideraciones finales del capítulo.

A partir de todo el análisis anterior puede llegarse a las siguientes conclusiones:

⋆ Entre las diversas plataformas de desarrollo, el *MATLAB* resulta el más completo por su paquete de procesamiento matemático para la estimación y validación de modelos

dinámicos así como por su comunicación en tiempo real a través de una tarjeta de adquisición de datos.

⋆ Las herramientas existentes en *MATLAB*, por su complejidad, no resultan viables para que los especialistas de SIMPRO realicen el proceso de identificación y síntesis del controlador para las plataformas de simulación por lo que se hace necesario crear unas a la medida.

⋆ La interfaz que se desarrolle debe permitir la excitación del sistema, en lazo cerrado, con una señal PRBS y una ganancia conocida en el controlador. Ambas cosas deben poder ser definidas por el usuario. También debe garantizar la estimación y validación del modelo paramétrico discreto que describa la dinámica de la planta a partir de las señales PRBS y posición medida. Por último, debe permitir la síntesis del controlador propuesto por GARP y la comprobación de su desempeño. Estos elementos constituyen los requerimientos de diseños para la interfaz.

CAPÍTULO 2
DISEÑO E IMPLEMENTACIÓN DE LA INTERFAZ

En el capítulo anterior se analizan las ventajas y desventajas de algunos *softwares* con vista de utilizar el más indicado para la creación de la interfaz de identificación y control de la plataforma de dos grados de libertad, concluyéndose que el *MATLAB* es la mejor alternativa. Este es un *software* interactivo donde se pueden crear aplicaciones propias, 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, 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).

El *MATLAB* cuenta con un conjunto de herramientas denominadas Interfaz Gráfica de Usuarios (GUI, *Graphical User Interface*) donde el diseño de una interfaz en este editor (Figura 2–1) crea dos ficheros con el nombre de la interfaz, uno con extensión .fig (Figura) y otro con extensión .m (*M-files*).

El archivo .fig contiene la descripción de los componentes que contiene la interfaz y el archivo .m es el código fuente del programa donde están las funciones y el *callback*, que se define como la acción que lleva a cabo un objeto del GUI cuando sea activado por el usuario.

Cuando se habla de objetos se está haciendo referencia a los botones cuadrados, ya sean de pulsos (*push button*) o de ON-OFF (*toggle button*), botones redondos (*radio button*),

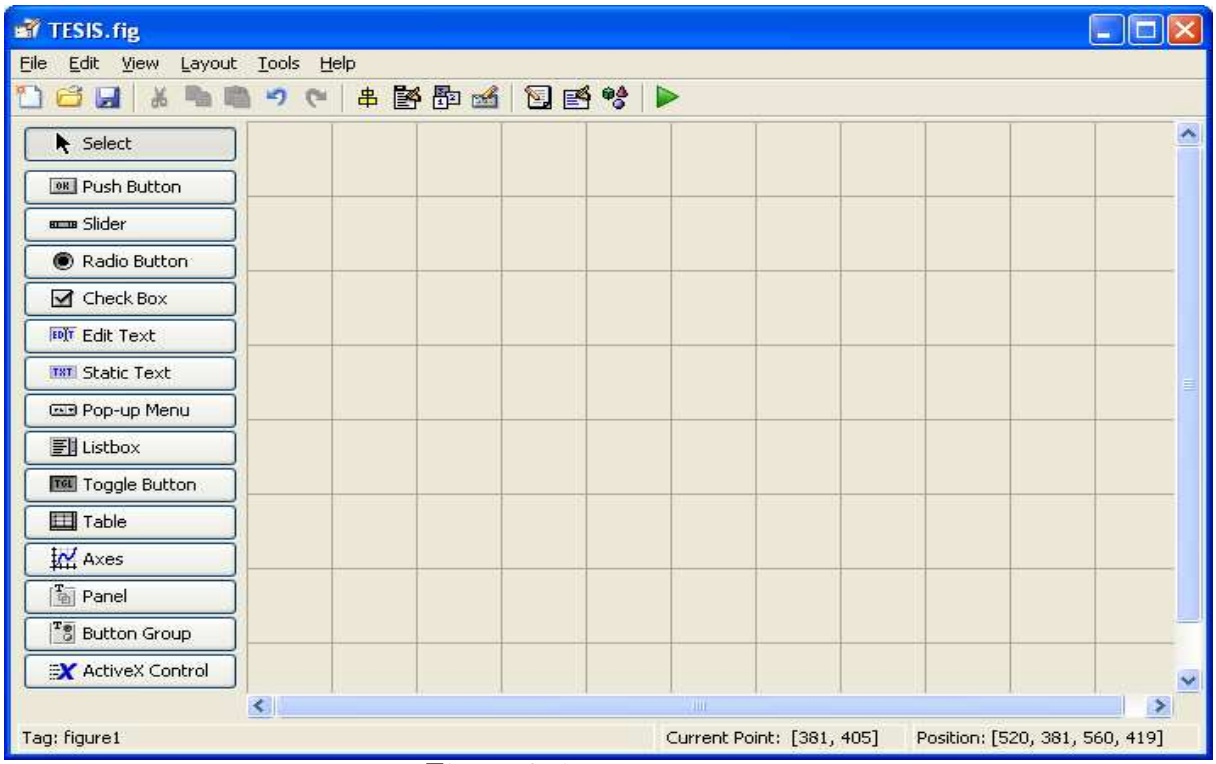


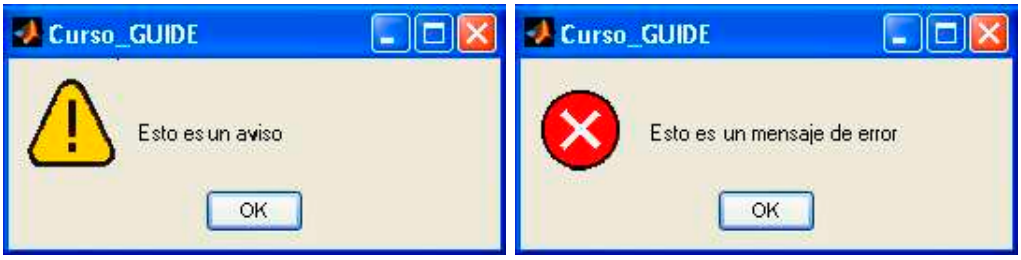
Figura 2–1: Editor del GUI.

los submenús desplegables (*pop-up menu*), los editores de texto (*text edit*), rectángulos gráficos (*axes*), entre otros.

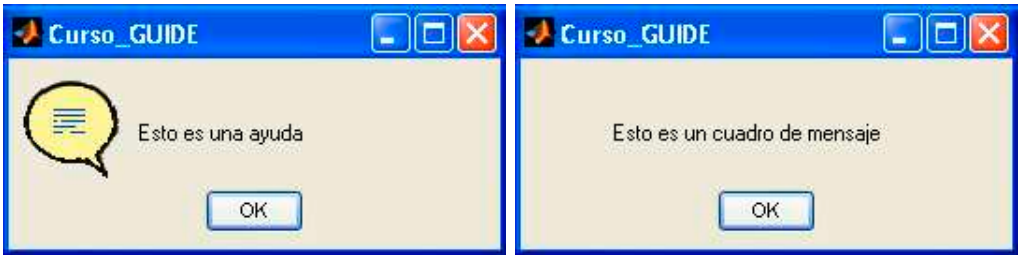
El GUI contiene diferentes tipos mensajes para el usuario entre los que se encuentran: aviso (Figura 2–2(a)), error (Figura 2–2(b)), ayuda (Figura 2–2(c)) y cuadro de mensajes (Figura 2–2(d)). También formas de interactuar con el usuario realizándole preguntas (Figura 2–3), que permiten guiar al usuario en el trabajo con la interfaz (Barragán, 2007).

En el caso especial de las preguntas se puede ejecutar o no "sentencias" dependiendo de la respuesta escogida.

El proceso de identificación es necesario para poder conocer el modelo real de la planta, pero por la complejidad de determinar analíticamente muchos de los parámetros del modelo, se recomienda determinar el modelo por métodos de identificación experimental (Rubio, 2009).



(a) Aviso (b) Error



(c) Ayuda (d) Mensajes o información

Figura 2–2: Mensajes del GUI.

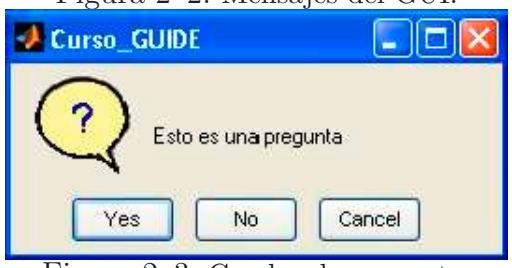


Figura 2–3: Cuadro de pregunta.

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).

Como la dinámica del sistema es claramente dependiente de la posición de la carga y el sistema es de tercer orden tipo uno, resulta conveniente identificarlo en lazo cerrado alrededor de diferentes posiciones. Esto no siempre es posible en el caso particular de la plataforma de dos grados de SIMPRO, pues por sus dimensiones y peso se hace inestable fácilmente. En tal caso se sugiere identificar alrededor de la posición central y cada articulación por separado (Rubio, 2009).

La entrada de excitación al sistema deben ser cuidadosamente elegidas de forma que los datos recogidos proporcionen toda la información posible sobre su dinámica. Para ello, conviene tener en cuenta los siguientes aspectos:

⋆ La señal de entrada debe tener componentes de frecuencias que exciten todo el ancho de banda del sistema. Por ejemplo, una señal sinusoidal 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, pues 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).

El algoritmo de identificación (Figura 2–4) implementado en la interfaz 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 su salida 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 hay que corregir antes de proceder a la estimación del modelo. Se trata, por tanto, de procesar los datos para facilitar y mejorar el proceso de estimación.

3. Elección de la estructura del modelo:

Si el modelo que se desea obtener es un modelo paramétrico, el primer paso es seleccionar 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:

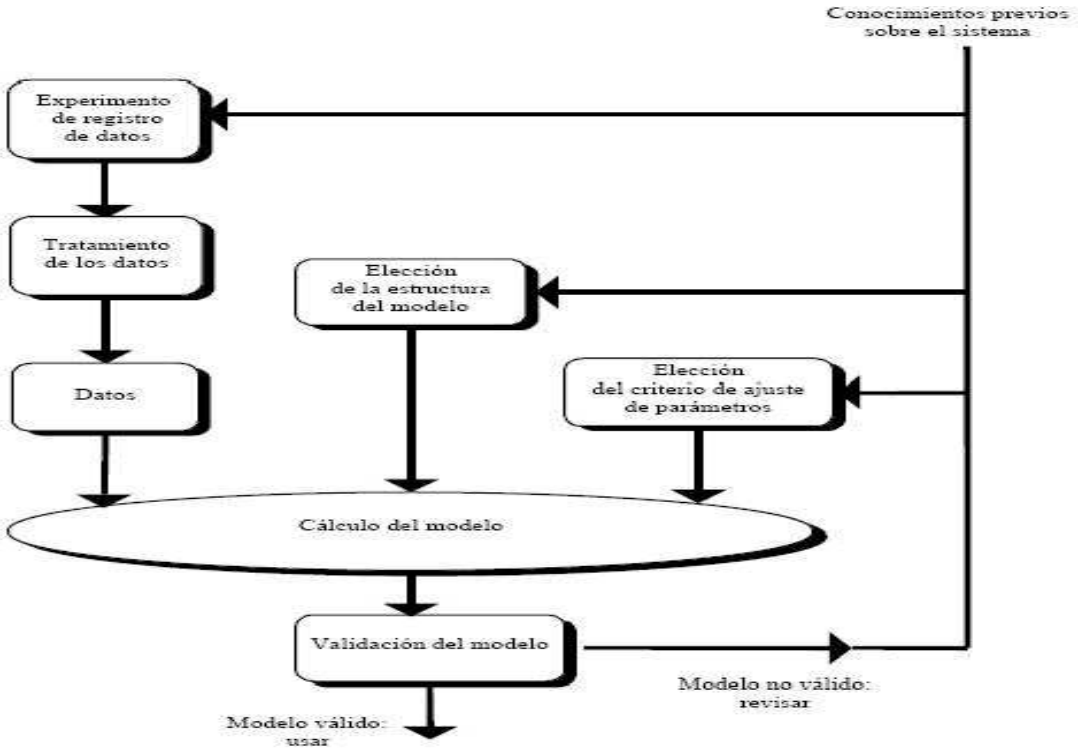


Figura 2–4: Proceso de identificación.

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

Según sea la causa estimada, deberá repetirse el proceso de identificación desde el punto correspondiente. Por tanto, el proceso de identificación es un proceso iterativo.

La interfaz de identificación y control implementada contiene tres interfaces realizadas con el GUI del *MATLAB*: **Identificación** (Figura 2–5), **Estimación** (Figura 2–8) y **Control** (Figura 2–9), las mismas permiten estimar y validar el modelo de la plataforma, a través de

este modelo sintetizar un controlador. En la interfaz **Identificación** esta implementado el primer paso del proceso de identificación y los restantes pasos en la interfaz **Estimación**, mientras que la interfaz **Control** es la encargada de la reducción del modelo estimado a tercer orden tipo uno y sintetizar el controlador PI en cascada con un filtro de segundo orden.

Las interfaces cuentan con unos botones de desplazamiento hacia las demás interfaces del *software*, es decir, que desde cualquier interfaz en la que el usuario se encuentre trabajando puede acceder a la que estime conveniente, solamente pulsando el botón con el nombre de la interfaz deseada, estos botones son: *IDENTIFICACION*, *ESTIMACION* y *CONTROL*.

Las interfaces tienen implementado un botón denominado *GUARDAR* que tiene como función brindarle al usuario la posibilidad de guardar los valores de las variables usadas y señales, modelos obtenidos, etc., en un fichero .mat con el nombre y en la dirección donde estime conveniente el usuario, después de terminado el experimento.

2.1. Implementación de la interfaz Identificación.

La interfaz **Identificación** (Figura 2–5), del proceso de identificación anteriormente descrito, tiene implementado la obtención de datos de entrada - salida, para lo cual la planta es excitada en la entrada con una señal PRBS y se registra el comportamiento a la salida.

De forma visual la interfaz cuenta con dos paneles principales *VARIABLES* y *RESULTADOS*. El panel *VARIABLES* cuenta a su vez con dos subpaneles *Señal de entrada* y *Plataforma*, los dos tienen sus variables propias, así de esta forma será más fácil para el usuario el manejo de dicha interfaz.

El botón de *COND INIC*, es el de condiciones iniciales. Cuando el programa es ejecutado por primera vez , o se pulsa este botón las variables toman valores que permiten comenzar el proceso de identificación de forma segura, pues son valores que en la práctica han dado buenos resultados.

Se necesitó la creación del fichero *tmpfile.mat* para la comunicación entre el *workspace* del *MATLAB* y el *SIMULINK*, además guarda los últimos valores de las variables usadas, que

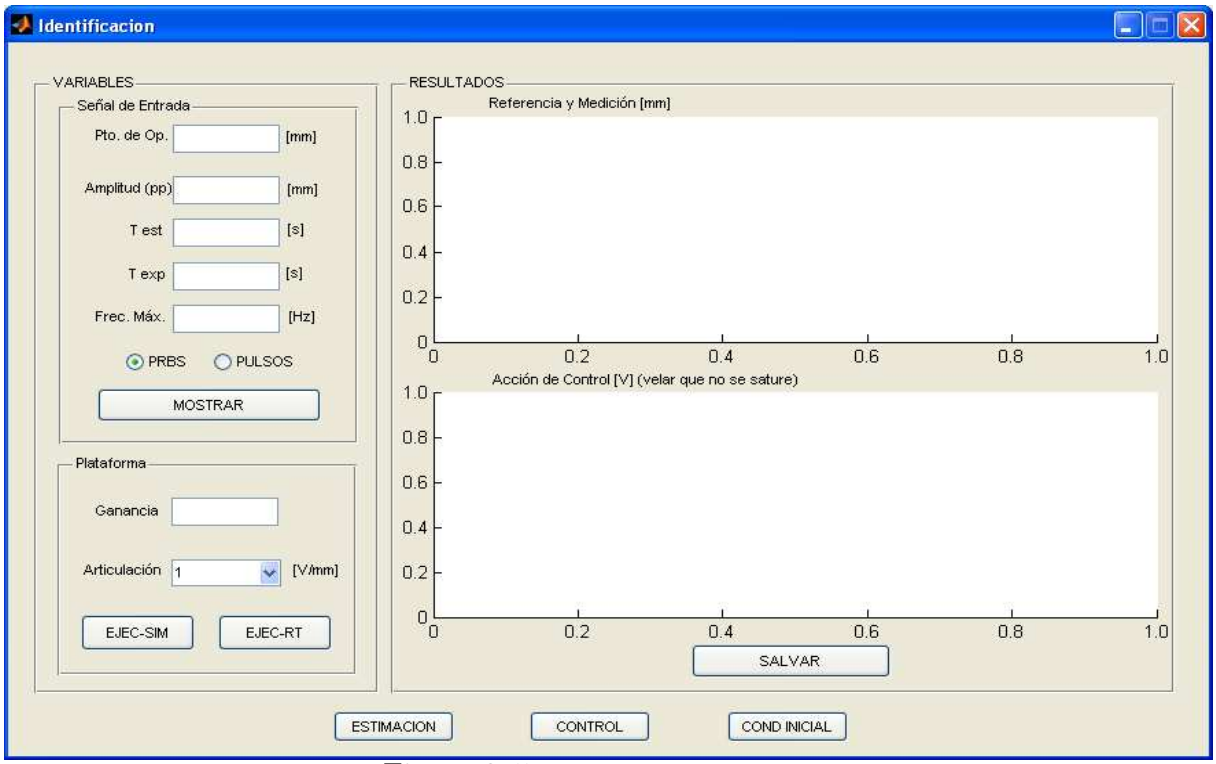


Figura 2–5: Interfaz Identificación.

serán cargados en la interfaz cuando arranque, de no existir el fichero la interfaz muestra las condiciones iniciales programadas.

2.1.1. Definición de la señal de excitación.

El diseño de la señal de excitación (PRBS) se lleva a cabo desde el subpanel *Señal de entrada* que contiene las variables:

⋆ Pto. de Op. (mm): La posición del punto de operación sobre el cual el sistema va a ser excitado para su posterior identificación, es bueno decir que el recorrido máximo de las articulaciones de la plataforma es de 320 mm, en el *software* se toma un rango de posición de $0 - 300$ mm, con dos zonas de seguridad $0 - 10$ mm y $290 - 300$ mm. Se le informa mediante un mensaje de aviso al usuario cuando se encuentra operando en la zona de seguridad y mediante un mensaje de error cuando está fuera del rango de operación.

⋆ Amplitud (pp): es la distancia pico a pico (pp) alrededor del punto de operación donde se describirá el comportamiento de la señal de excitación. Matemáticamente hablando: $A_{máx} = PO - A/2$ y $A_{mín} = PO - A/2$ donde $A_{máx}$ es la amplitud máxima y

$A_{mín}$ es la amplitud mínima, PO el punto de operación y A es la amplitud introducida por el usuario. Se le informa al usuario con mensajes de errores si $A_{mín}$ y $A_{máx}$ se encuentran fuera del rango de operación y con mensajes de avisos si están en la zona de seguridad.

⋆ Test (s): es el tiempo de experimento estable, en el que las articulaciones se sitúan en el punto de operación indicado.

⋆ Texp (s): es el tiempo de experimento, después del Test, se comienza a exitar el sistema durante este tiempo.

⋆ Frec. Máx. (Hz): la frecuencia máxima de la señal de excitación.

Luego de establecer el valor de las variables, para generar la señal de entrada se pulsa el botón *MOSTRAR*, que es el encargado de ejecutar la función PRBS, implementada en el código fuente de la interfaz **Identificación**, que genera la señal de excitación. Esta función devuelve la señal PRBS en un vector formado por los valores de t y u.

2.1.2. Selección de la articulación a identificar.

En el subpanel *Plataforma* se define la ganancia del controlador proporcional utilizado para la identificación y, con un menú desplegable, la articulación de la plataforma a la cual desea identificar. Este subpanel también cuenta con dos botones: *EJEC-SIM* y *EJEC-RT*. El botón *EJEC-RT* hace un llamado al fichero *mdlRT.mdl* (Figura 2–6) del *SIMULINK* que mediante el *Real Time Workshop* de *MATLAB/SIMULINK* se conecta a la plataforma en tiempo real, usando una tarjeta de adquisición de datos *HUMUSOFT MF614*, la excita con la PRBS diseñada y recoge las variaciones de su posición.

El botón *EJEC-SIM* simula el comportamiento de la plataforma a partir de un modelo previamente obtenido, colocado en el fichero *mdltest.mdl* (Figura 2–7). Esta opción permite trabajar con la interfaz sin necesidad de estar conectados a la planta real.

2.1.3. Respuesta del sistema.

El panel *RESULTADOS* contiene dos gráficos: *Referencia y Medición* y *Acción de control*. En el primero se muestra la señal de excitación después de ejecutar *MOSTRAR* y luego

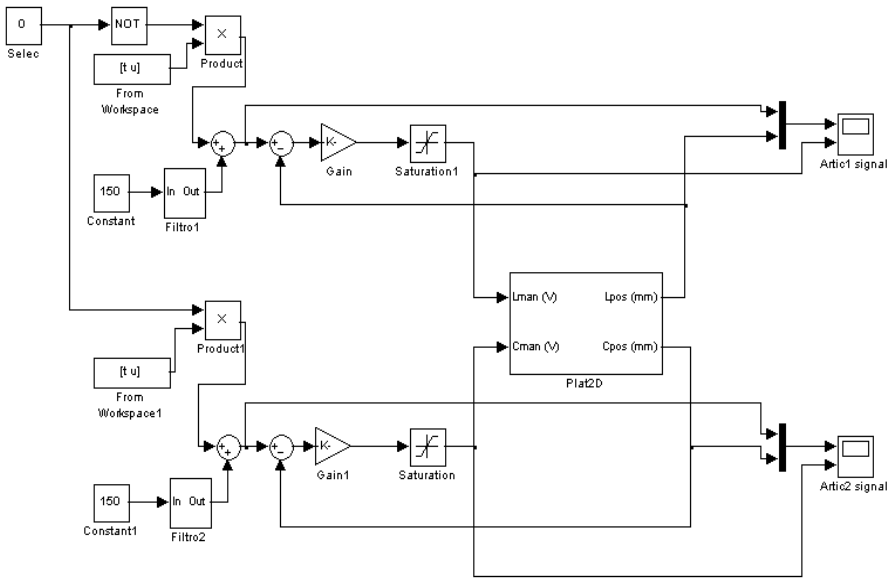


Figura 2–6: Modelo de ejecución de la identificación en tiempo real.

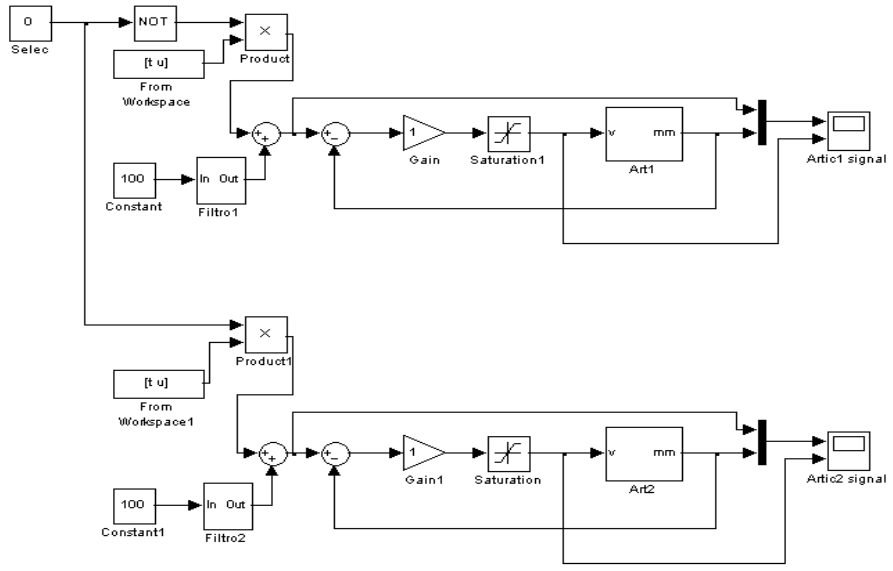


Figura 2–7: Modelo de ejecución de la identificación simulada.

"si se activa cualquiera de los dos botones de ejecución del subpanel *Plataforma*" este muestra, además de la señal de excitación, la señal medida a la salida. El segundo muestra la acción de control aplicada a la válvula. Una vez realizado el experimento, el usuario debe chequear visualmente que el sistema haya respondido apreciablemente a la señal de excitación pero que no haya llegado a los extremos. También debe chequear visualmente, que la acción de control no haya quedado saturada en ningún momento. Esto garantiza que la no linealidad no esté presente en la respuesta del sistema.

Este panel cuenta además con un botón de *GUARDAR*, que "entre otras funciones" le brinda la posibilidad al usuario de guardar las señales de excitación del sistema y la medida a la salida en un fichero .mat. Este fichero es el que utiliza la siguiente interfaz para estimar el modelo.

2.2. Implementación de la interfaz Estimación.

La interfaz **Estimación** (Figura 2–8) tiene implementado los restantes pasos del proceso de identificación: tratamiento previo de los datos registrados, elección de la estructura, estimación del modelo y validación del mismo.

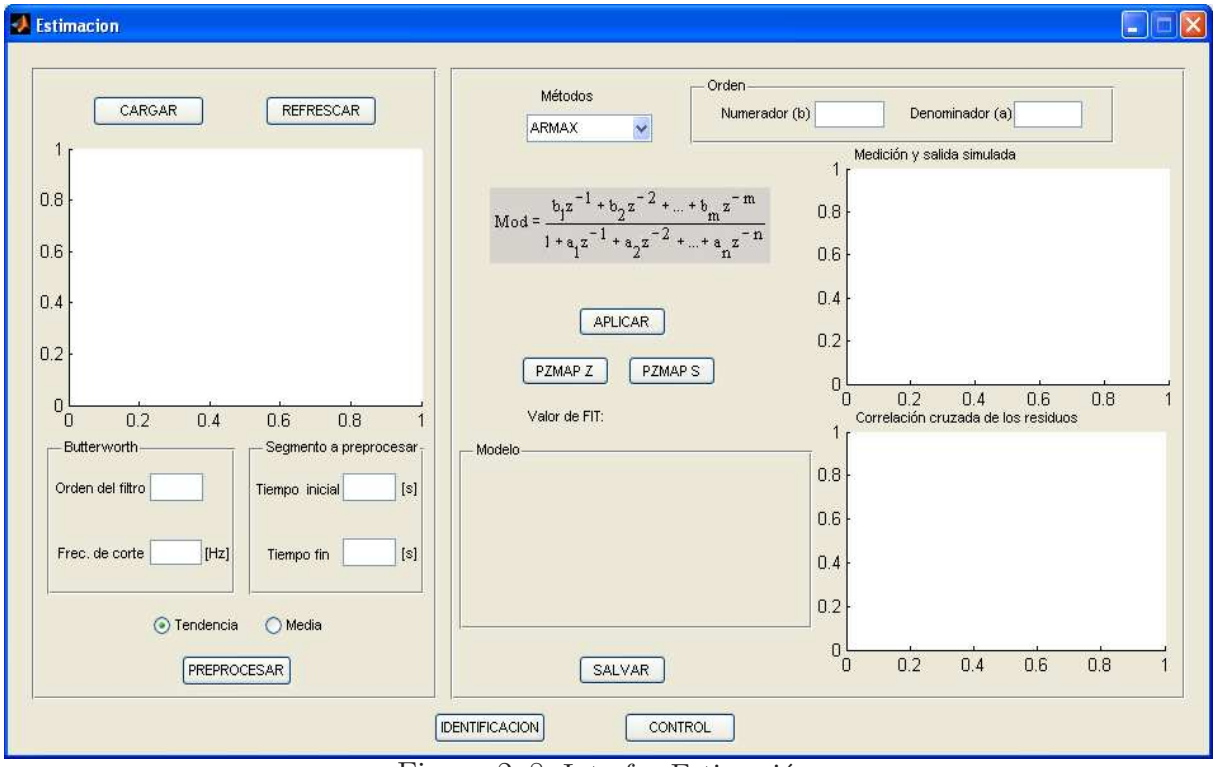


Figura 2–8: Interfaz Estimación.

Al inicializar la interfaz, mediante el botón *CARGAR* se muestran las señales de entrada y de salida de la planta cargando el fichero .mat en el cual fueron salvadas las variables del proceso de identificación, aunque si se proviene de la interfaz **Identificación**, se muestran estas señales automáticamente.

El usuario "en esta interfaz" obtiene el modelo discreto estimado en lazo cerrado "después de realizar varios experimentos" hasta obtener el deseado. A través de este "conociendo la ganancia del controlador" se despeja el modelo en lazo abierto y se lleva a continuo.

2.2.1. Preprocesamiento de la señal.

Los datos registrados están generalmente acompañados de ruidos indeseados que se corrigen mediante un filtro *butterworth* del orden y frecuencia de corte que establezca el usuario. Se selecciona el segmento de la señal a preprocesar, que va a comenzar en el tiempo de inicio hasta el tiempo final. El preprocesamiento además consiste en eliminarle las medias o las tendencias a la señal de salida. Todo esto se ejecuta al pulsar el botón *PREPROCESAR*.

2.2.2. Selección de la estructura y orden del modelo a estimar.

El modelo paramétrico que se va a obtener, va a estar dado por una función transferencial en tiempo discreto de la forma:

$$Mod = \frac{b_1 z^{-1} + b_2 z^{-2} + ... + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n}} \tag{2.1}$$

donde b_1, $b_2 ... b_m$, son los coeficientes del numerador y a_1, $a_2 ... a_n$, son los coeficientes del denominador. El orden del numerador nunca va a exceder el orden del denominador y este a su vez no va a ser menor que 3.

La ecuación genérica que devuelve los modelos paramétricos ARMAX, ARX, OE y BJ, implementados en el menú desplegable es de la forma:

$$A(z, \theta_A) \cdot y(k\tau) = \frac{B(z, \theta_B)}{F(z, \theta_F)} \cdot u(k\tau) + \frac{C(z, \theta_C)}{D(z, \theta_D)} \cdot e(k\tau) \tag{2.2}$$

⋆ ARX (*Auto Regressive and Exogenous Variable*).

Estructura auto-regresiva $[A(z) \cdot y(k)]$ con variables exógenas $[B(z) \cdot u(k)]$, suponiendo un ruido blanco de media cero y varianza constante. Se resuelve directamente por el algoritmo de mínimos cuadrados, que responda a la forma:

$$A(z) \cdot y(k) = B(z) \cdot u(k - nk) + e(k) \tag{2.3}$$

⋆ 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:

$$A(z) \cdot y(k) = B(z) \cdot u(k - nk) + C(z) \cdot e(k) \tag{2.4}$$

⋆ 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:

$$y(k) = \frac{B(z)}{F(z)} \cdot u(k - nk) + e(k) \tag{2.5}$$

⋆ BJ (*Box-Jenkins*).

Estructura auto-regresiva con variables exógenas, cuya parte determinista no tiene coeficientes 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:

$$y(k) = \frac{B(z)}{F(z)} \cdot u(k - nk) + \frac{C(z)}{D(z)} \cdot e(k) \tag{2.6}$$

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).

2.2.3. Presentación de los resultados para la validación del modelo.

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. Se han implementado varias herramientas de las que el usuario puede auxiliarse para realizar una validación del modelo entre las que se encuentran:

⋆ El análisis de la desviación estándar de cada coeficiente de la función transferencial obtenidos y sus variaciones son mostrados en textos estáticos, porque si alguna de las variaciones es mayor que su coeficiente este se desprecia.

⋆ Porciento de convergencia entre la medición y la salida simulada (FIT).

Se muestra en un texto estático el porciento de convergencia entre la medición y la salida simulada que mientras más se acerque este valor a 100 el modelo estimado va a tener una respuesta similar al real.

⋆ Correlación cruzada de los residuos.

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

$$\varepsilon(t) = \varepsilon(t, \theta) = y(t) - y_e(t, \theta) \tag{2.7}$$

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 $\varepsilon(t)$ 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:

$$R_{\varepsilon u} = \frac{1}{N} \sum_{t-1}^{N} \varepsilon(t + \Pi) \cdot u(t) \tag{2.8}$$

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).

⋆ Posición de los polos y ceros.

En los botones *PZMAP S* y *PZMAP Z* se pueden apreciar la ubicación de los polos y ceros, en el *PZMAP Z* del modelo discreto estimado en lazo cerrado y en el *PZMAP S* del modelo continuo en lazo abierto, despejado del modelo discreto.

El botón *GUARDAR*, tiene como función salvar los modelos obtenidos, junto con otros datos de interés, como los valores de las variables empleadas, para su uso en otro momento. Luego el usuario puede sintetizar el control para la articulación seleccionada cambiando a la siguiente interfaz **CONTROL**.

2.3. Implementación de la interfaz Control.

La interfaz **Control** (Figura 2–9) realiza la síntesis del controlador mediante el modelo en lazo abierto calculado anteriormente y se testea su desempeño en tiempo real.

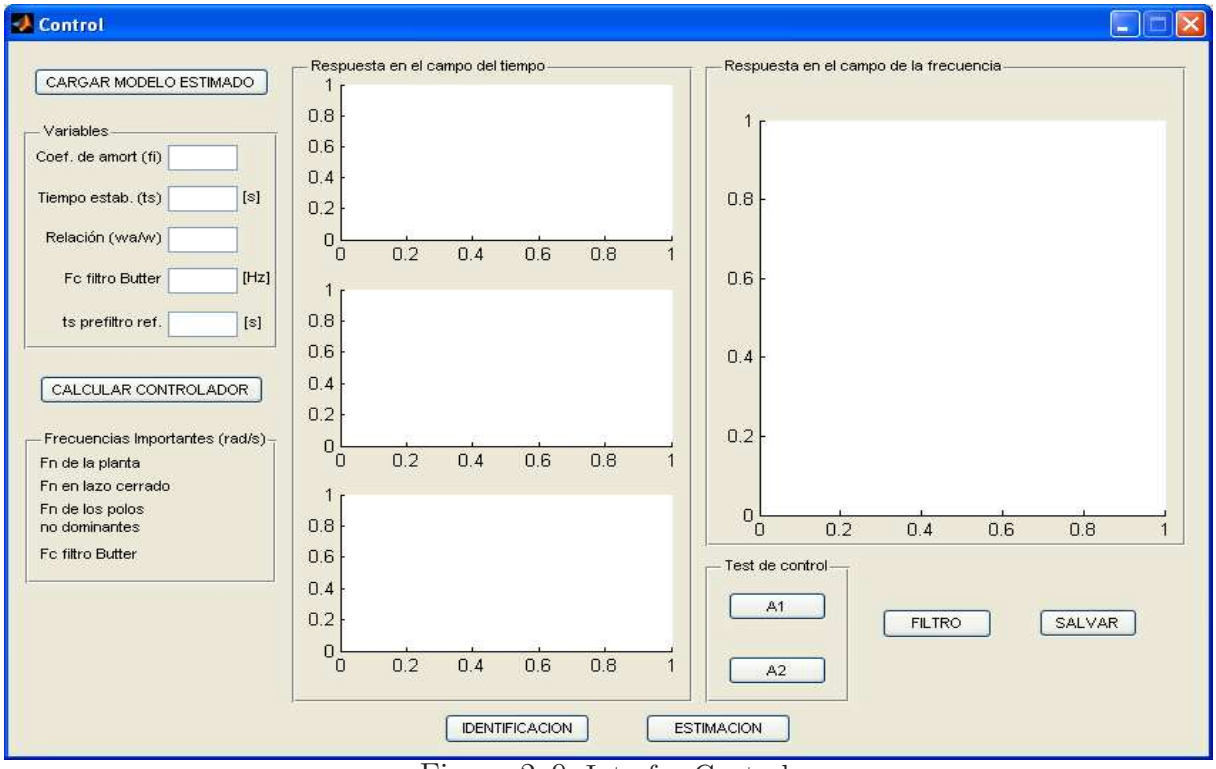


Figura 2–9: Interfaz Control.

Al inicializar la interfaz mediante el botón *CARGAR MODELO ESTIMADO* se carga el fichero .mat en el cual fueron salvadas las variables del proceso anterior, entre ellas el modelo continuo en lazo abierto, aunque si se proviene de la interfaz **Estimación** (Figura 2–8), estas variables son cargadas automáticamente y se presentan las condiciones iniciales programadas de los índices de desempeño.

2.3.1. Establecimiento de los índices de desempeño.

Entre los índices de desempeño programados se encuentran: el coeficiente de amortiguamiento (φ), tiempo de establecimiento (t_s), relación entre la frecuencia de los polos no dominantes y el dominante ω_a/ω, frecuencia de corte (f_c) del filtro y tiempo de establecimiento del prefiltro (t_{sp}).

Para comprobar el desempeño del controlador se utiliza un prefiltro de segundo orden después de la señal de excitación para garantizar que la entrada sea una señal suave como las provenientes de un supuesto mundo virtual. El prefiltro es un sistema de segundo orden típico que responde a la forma:

$$FILTRO = \frac{\omega_n^2}{s^2 + 2\varphi\omega_n s + \omega_n^2} \tag{2.9}$$

donde el valor de ω_n es calculado como: ω_n=$4/\varphi t_{sp}$.

Las válvulas proporcionales de flujo *FESTO MPYE-5-3/8* usadas en la plataforma, tienen un ancho de banda de 40 Hz. Para prevenir que el mando dado por el controlador no presente frecuencias superiores, la interfaz cuenta con un filtro *butterworld* de segundo orden, después del controlador, al cual el usuario le establece su frecuencia de corte.

2.3.2. Síntesis del controlador y presentación de su respuesta teórica.

Presionando el botón *CALCULAR CONTROLADOR* se procede a determinar el controlador como el mismo indica de la siguiente forma:

Representando la planta identificada de la forma:

$$A(s)Y(s) = B(s)U(s) \tag{2.10}$$

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

$$R(s)U(s) = S(s)(Y_d(s) - Y(s)) \tag{2.11}$$

donde R y S son los polinomios del controlador y $Y_d(s)$ la señal de entrada o referencia. Bajo estas condiciones es posible establecer que si la respuesta dinámica deseada, $Y_m(s)$,

ante la señal $Y_d(s)$, está descrita por:

$$A_m(s)Y_m(s) = B_m(s)Y_d(s) \tag{2.12}$$

donde A_m y B_m son los polinomios del modelo en lazo cerrado deseado, entonces:

$$\frac{S(s)B(s)}{A_m(s)R(s) + S(s)B(s)} = \frac{S(s)B(s)}{A_c(s)} = \frac{B_m(s)}{A_m(s)} \tag{2.13}$$

donde A_c es la ecuación característica del sistema. Los polinomios R y S se obtienen resolviendo la ecuación diofántica correspondiente:

$$A_m(s)R(s) + S(s)B(s) = A_m(s) \tag{2.14}$$

La función de transferencia constituirá la planta base para el diseño.

$$\frac{B(s)}{A(s)} = \frac{k_m a_0}{s(s^2 + a_1 s + a_0)} \tag{2.15}$$

La obtención de los índices de desempeño simplifica significativamente si se logra compensar, mediante un filtro de segundo orden, los polos complejos conjugados de la planta y el par de polos de lazo cerrado que introduce este filtro deben ser no dominantes. De modo que se llega el controlador S/R propuesto:

$$\frac{S(s)}{R(s)} = \frac{k_p(s + k_i)(s^2 + a_1 s + a_0)}{s(s^2 + k_1 s + k_0)} \tag{2.16}$$

Para la planta con el controlador, se tiene como ecuación característica del sistema:

$$A_c(s) = s^2(s^2 + k_1 s + k_0) + k_m a_0 k_p(s + k_i) \tag{2.17}$$

Como ecuación característica deseada se propone:

$$A_m(s) = (s^2 + 2\varphi\omega s + \omega^2)(s + \omega_a)^2 \tag{2.18}$$

Manipulando algebraicamente las ecuaciones 2.17 y 2.18 e igualando los coeficientes se obtiene el sistema de ecuaciones que permite el cálculo de los parámetros del controlador k_p, k_i, k_1 y k_0 a partir de fijar φ, ω y ω_a según los índices de desempeño deseados

Disminuir el tiempo de establecimiento (t_s) para que el sistema siga mejor la referencia, hace que el controlador pueda llegar a tener valores por encima de cero decibel para las altas frecuencias lo cual puede ser perjudicial para las válvulas e implica el uso del filtro de *butterworth* después del cálculo del mando. (Rubio, 2011).

La interfaz cuenta con dos paneles donde se muestran las respuestas teóricas del sistema continuo en el campo del tiempo y de la frecuencia. En los gráficos del panel de *Respuesta en el campo del tiempo*, se muestran las respuestas en lazo cerrado ante entrada paso: en el primero, del modelo típico de segundo orden (Ecuación 2.9) y además la del controlador con el modelo continuo estimado, con y sin filtro *butterworld* para la señal del mando, en el segundo la del mando y en el tercero ante perturbaciones.

En el gráfico del panel de la *Respuesta en el campo de la frecuencia* se muestra la respuesta en el campo de la frecuencia en lazo abierto: del controlador solo y del controlador con el modelo continuo estimado en lazo abierto, así como su margen de fase y la ganancia.

El botón *ON-OFF*, denominado *FILTRO* al activarse muestra las respuestas antes mencionadas en el campo de la frecuencia, con el filtro de *butterworld* detrás del controlador, al desactivarse muestran las gráficas sin el efecto del filtro. Debe notarse que la inserción del filtro puede deteriorar el margen de ganancia y fase del sistema.

2.3.3. Comprobación del desempeño del controlador en la planta real.

En el panel *Test de Control* están implementados dos botones *A1* y *A2* con los cuales se realiza la comprobación del controlador calculado. En el botón *A1* se testea el control en la articulación uno y en el botón *A2* en la articulación dos. Al presionar uno de estos botones es abierto el *mdlControlReal* (Figura 2–10) en el que el usuario puede seleccionar mediante un interruptor la señal de excitación aplicada a la referencia. Las señales de excitación usadas son: una onda sinusoidal y una señal escalonada.

2.4. Consideraciones finales del capítulo.

Luego del diseño e implementación de la interfaz se llegan a las siguientes conclusiones:

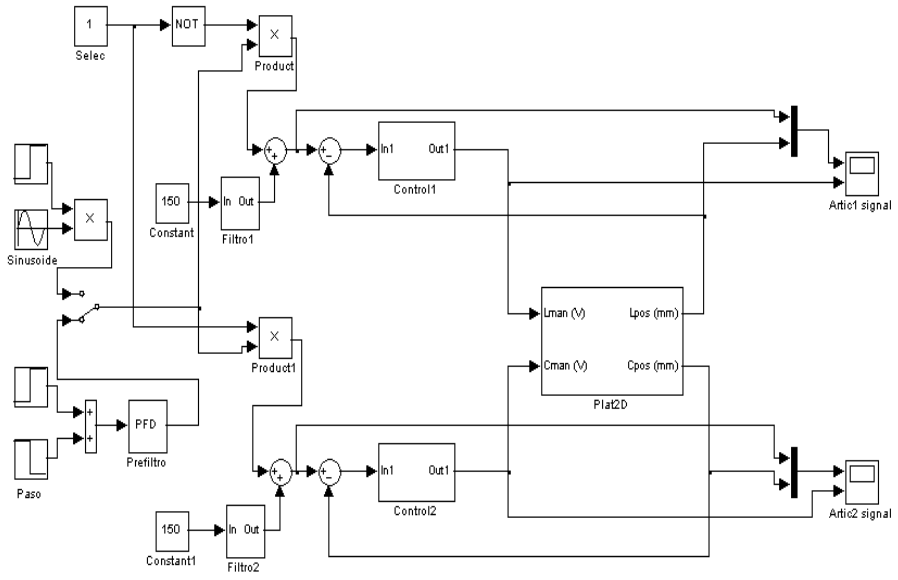


Figura 2–10: Modelo de ejecución de la identificación en tiempo real.

⋆ Tras el proceso de excitación de la planta con la señal PRBS, el usuario debe chequear visualmente que el sistema haya respondido apreciablemente a la señal de excitación pero que no haya llegado a los extremos. También debe chequear visualmente, que la acción de control no haya quedado saturada en ningún momento.

⋆ El modelo linealizado de un sistema electro neumático es de tercer orden tipo uno pero en este caso, la planta en cuestión, verdaderamente es multivariable y acoplada por lo que modelos de orden superior pueden representar mejor la dinámica real.

⋆ Disminuir el tiempo de establecimiento (t_s) para que el sistema siga mejor la referencia, hace que el controlador pueda llegar a tener valores por encima de cero decibel para las altas frecuencias y en tal caso es imprescindible utilizar un filtro del mando para proteger las válvulas, pero este entonces deteriora el margen de fase y ganancia del sistema. Por ello hay que establecer bien el compromiso entre t_s y f_c.

CAPÍTULO 3
APLICACIÓN Y VALIDACIÓN DE LA INTERFAZ

En el siguiente capítulo se presenta un informe detallado de los resultados de la interfaz implementada en una prueba en tiempo real realizada al simulador de conducción de dos grados de libertad, en el que se llevaron a cabo los procesos de identificación y síntesis del controlador.

3.1. Proceso de identificación.

El proceso de identificación debe hacerse en lazo cerrado porque de otra forma no podría lograrse variaciones estables alrededor de cada tramo que se defina dado que el sistema tiene un integrador implícito. El lazo se cierra con un controlador proporcional, para no alterar el orden del sistema.

Se determinó un tiempo de establecimiento de 10 *seg* para situar las articulaciones de la plataforma en 150 *mm*, la mitad del recorrido permitido, excitándose después la articulación uno, mientras que la articulación dos se mantiene en reposo. Usando una PRBS de amplitud 70 *mm* (*pp*) y 0,5 Hz de frecuencia, como señal de excitación durante un tiempo de 40 *seg*, se mantuvo al sistema persistentemente excitable, requisito indispensable para lograr una buena identificación.

Se muestran en los paneles de la interfaz **Identificación** (Figura 3–1) los valores de las variables generales, en el panel *Señal de entrada*, los parámetros de la señal de excitación PRBS escogida, en el panel *Plataforma* la articulación uno seleccionada y la ganancia 0,019 del controlador proporcional.

Se aprecian también en los rectángulos gráficos las señales: de excitación (azul) y de medición de la posición de la articulación uno (verde), en el primero y en el segundo la

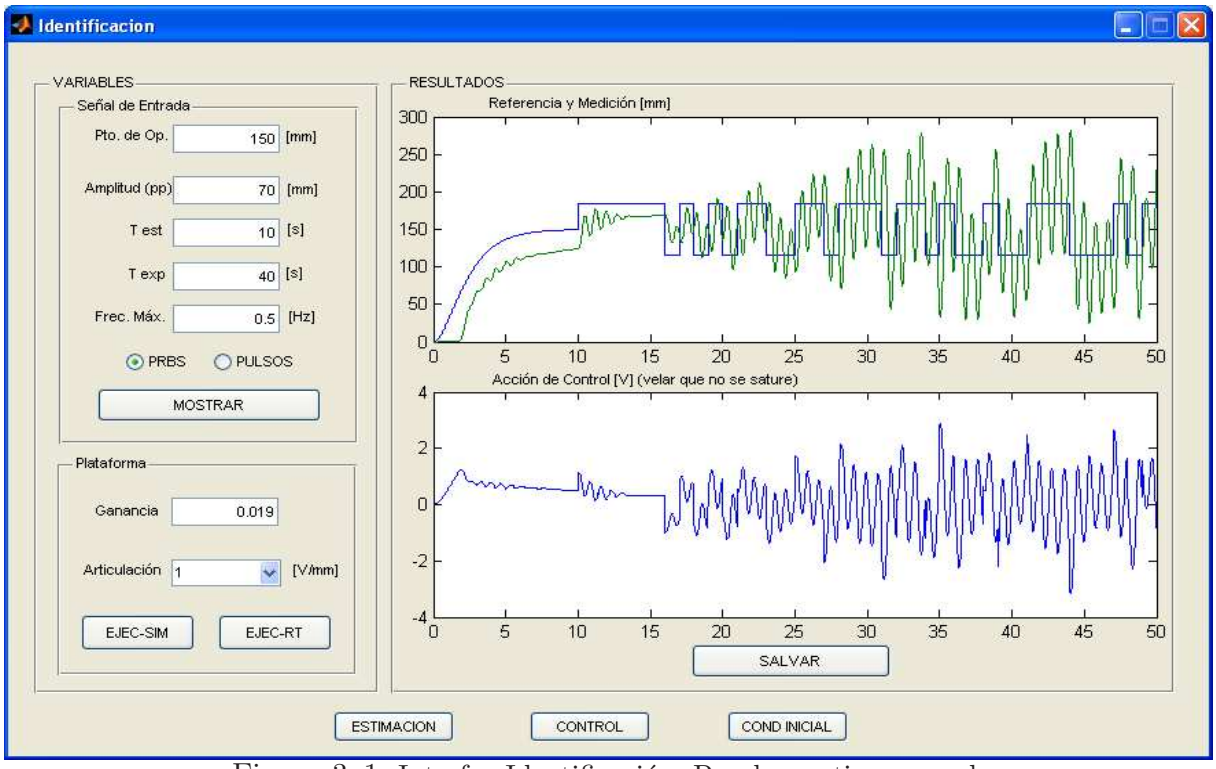


Figura 3–1: Interfaz Identificación. Prueba en tiempo real.

señal del mando a la válvula (azul). La señal de salida del controlador es de $\pm 5\ V$, se puede apreciar que este no presenta saturación porque se mantiene la señal por debajo de esos valores.

3.2. Proceso de estimación.

En la Figura 3–2 se puede apreciar el proceso de estimación del modelo, a continuación se realiza una descripción detallada del mismo, conjunto con los pasos a seguir.

Al iniciar la interfaz **Estimación** (Figura 2–8) si se proviene de la interfaz **Identificación** (Figura 2–5), se muestran las señales: de PRBS y la medición de salida en el rectángulo gráfico, sino, se deben cargar estos datos del fichero .mat salvado en la interfaz **Identificación**.

De la señal representada se selecciona el segmento con el cual se va a estimar el modelo de la plataforma, se toma el segmento desde 30 *seg* hasta 50 *seg*. Se filtra la señal de medición con un filtro *butterworth* de 3 orden y 10 Hz de frecuencia de corte para eliminar los ruidos en la medición. Al segmento seleccionado se le eliminaron las tendencias como se muestra en la interfaz **Estimación** (Figura 3–2).

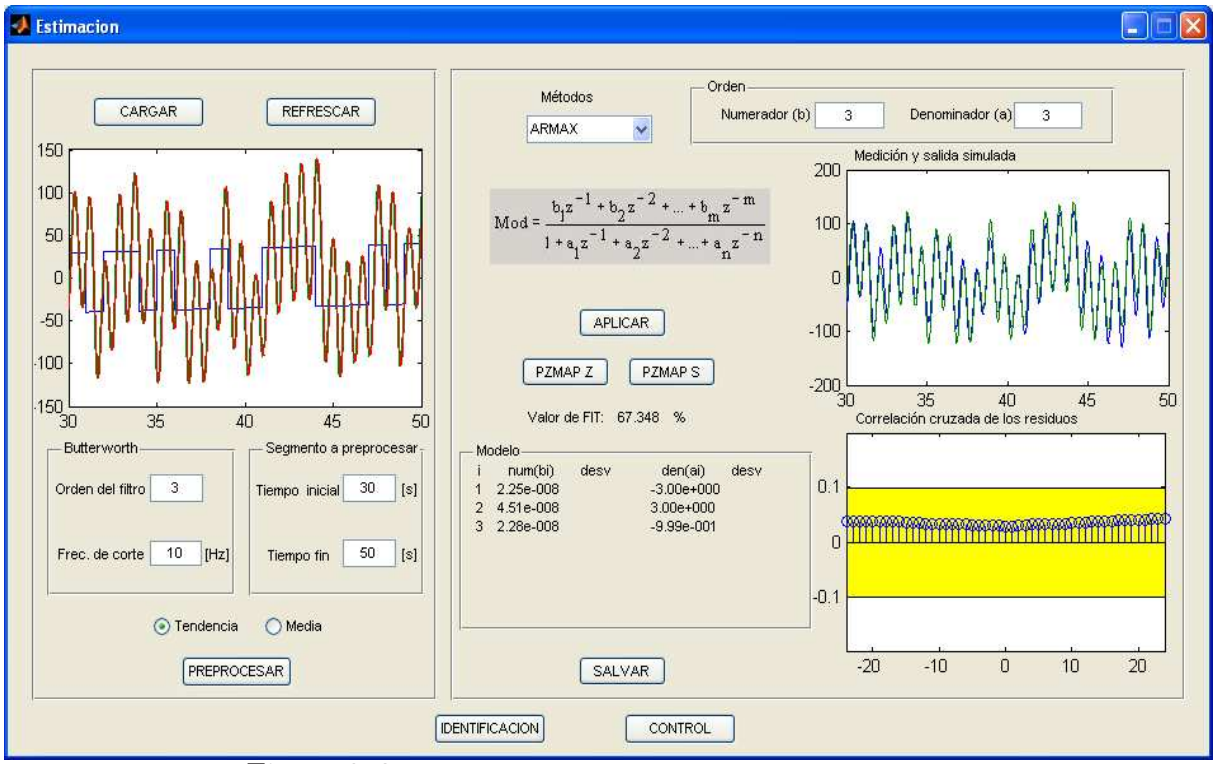


Figura 3–2: Interfaz Estimación. Prueba en tiempo real.

Mediante el método ARMAX se obtiene el modelo en lazo cerrado:

$$Mod = \frac{10^{-8}(2{,}25z^2 + 4{,}45z + 2{,}20)}{z^3 - 3{,}00z^2 + 3{,}00z - 0{,}99} \tag{3.1}$$

este modelo es de orden 3, para una mayor compresión el modelo es expresado en potencia positivas de z, pero en la interfaz se muestra en potencias negativas de z, así como los valores de los coeficientes con sus respectivas varianzas. Si alguna de estas varianzas es modularmente mayor que el coeficiente correspondiente, este se desprecia y el modelo a estimar puede que sea de un orden inferior al indicado, entonces se procede a estimar el modelo de nuevo variando el orden.

El modelo estimado tiene un FIT de 67,3 %, este valor representa el porciento de convergencia entre la medición (verde) y la salida simulada (azul) mostradas en el rectángulo gráfico correspondiente, así como también se observa la correlación cruzada de los residuos y estos se encuentran dentro de la franja amarilla, por lo que no tienen problemas.

Al observar la posición de los polos y ceros del modelo discreto (Figura 3–3), pulsando el botón *PZMAP Z* se aprecia que no hay cancelación entre ellos.

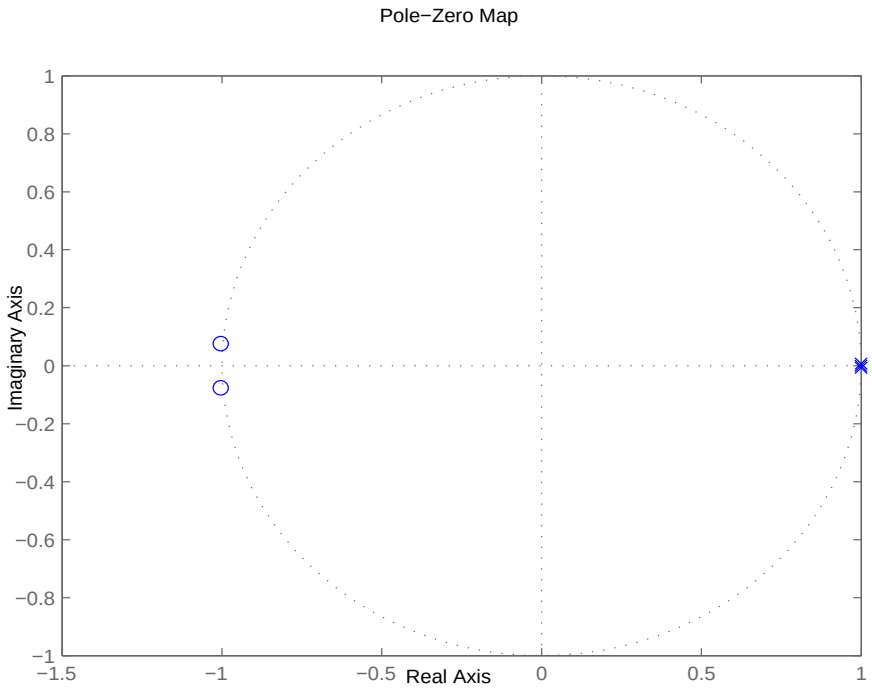


Figura 3–3: Posición de los polos y ceros del modelo discreto.

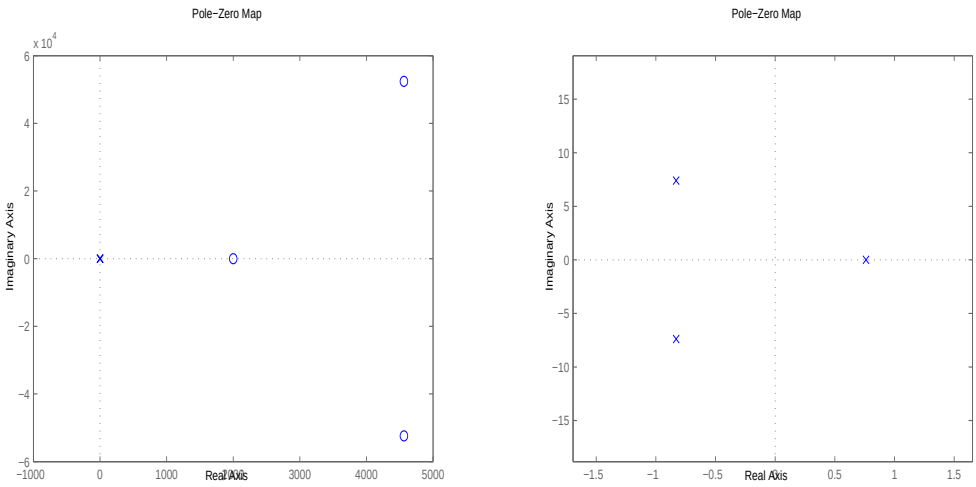
Luego de estimado y validado el modelo discreto en lazo cerrado, conociendo la ganancia del controlador se despeja el modelo continuo en lazo abierto obteniéndose:

$$Mod = \frac{-8{,}577 \cdot 10^{-10}s^3 + 9{,}543 \cdot 10^{-6}s^2 - 2{,}388s + 4744}{s^3 + 0{,}899s^2 + 54{,}12s - 42{,}14} \tag{3.2}$$

del cual se muestra la posición de los polos y ceros (Figura 3–4(a)), mediante el *PZMAP S*, donde se observa que no hay cancelación entre los polos y los ceros del modelo.

Al aumentar la resolución de la Figura 3–4(a) en la posición de los polos, se puede ver que hay dos polos complejos conjugados dominantes y que el restante polo se encuentra muy próximo al origen como lo expresa la Figura 3–4(b).

Al presionar el botón *GUARDAR* se brinda al usuario la opción de guardar los modelos continuo y discreto, así como el segmento preprocesado de la señal proveniente de la interfaz **Estimación** (Figura 3–2), en un fichero .mat.



(a) Posición de los polos y ceros (b) Posición de los polos dominantes

Figura 3–4: Modelo continuo.

3.3. Síntesis y validación del controlador.

Para realizar la síntesis del controlador mediante el modelo estimado, en la interfaz **Control** (Figura 3–5) se dejaron como valores de los índices de desempeño los programados en las condiciones iniciales, que son: coeficiente de amortiguamiento de 1 y un tiempo estable del experimento de 1 *seg*, buscando obtener una respuesta rápida y sobreamortiguada, la relación de la frecuencias de los polos del controlador de 10 veces alejado uno del otro, un tiempo de establecimiento para el prefiltro de 1 *seg* y una frecuencia de corte de 30 Hz para el filtro *butterworth* de segundo orden en vista de prevenir que el mando dado por el controlador presente frecuencias superiores a las válvulas proporcionales de flujo *FESTO MPYE-5-3/8* usadas en la plataforma, que tienen un ancho de banda de 40 Hz.

Al presionar el botón *CALCULAR CONTROLADOR* de la Figura 3–5 se sintetiza el controlador de la siguiente forma:

El modelo continuo en lazo abierto estimado (Ecuación 3.2) se reduce al modelo de tercer orden tipo uno de la plataforma:

$$Modred = \frac{4744}{s^3 + 1{,}66s^2 + 55{,}38s} \tag{3.3}$$

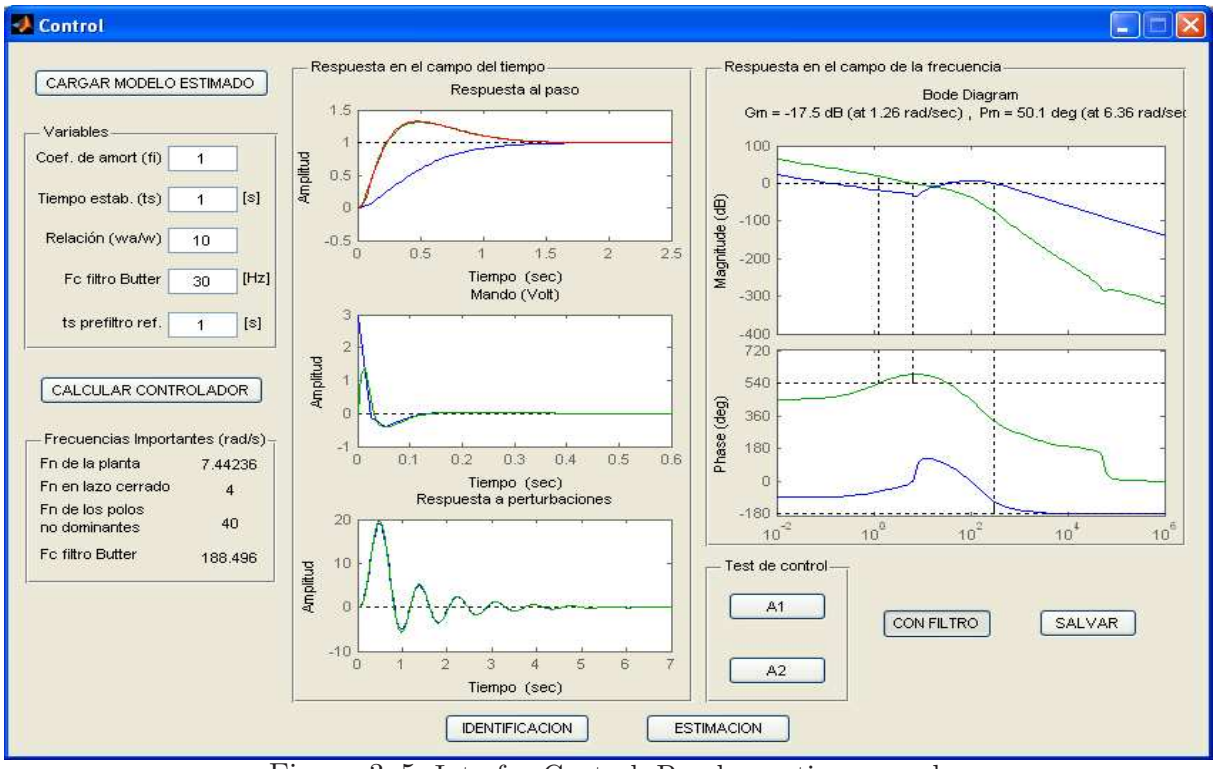


Figura 3–5: Interfaz Control. Prueba en tiempo real.

hacer notar que el polo que se encontraba cerca del origen (Figura 3–4(b)) ahora se encuentra en el origen. A partir de este modelo reducido se obtiene el controlador (Ecuación 3.4).

$$C_n = \frac{2{,}481s^3 + 9{,}534s^2 + 1491s + 254{,}4}{s^3 + 88s^2 + 2256s} \tag{3.4}$$

El filtro *butterworth* implementado en este caso quedó diseñado como se muestra a continuación:

$$F_n = \frac{355530}{s^2 + 266{,}6s + 355530} \tag{3.5}$$

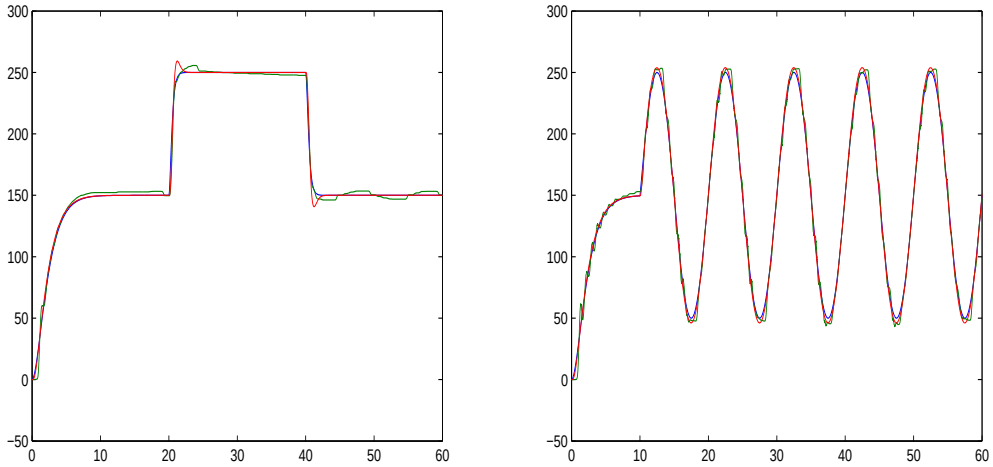
Se visualizan valores de las frecuencias de interés del sistema como son: frecuencias naturales de la planta 7,44 rad/seg, en lazo cerrado 4 rad/seg y de los polos no dominantes 40 rad/seg.

En los rectángulos gráficos del panel *Respuesta en el campo del tiempo* se pueden apreciar las respuestas ante entrada paso del modelo típico (Ecuación 2.9), del sistema continuo estimado (Ecuación 3.2) con filtro (Ecuación 3.5) y sin filtro, además de la respuesta ante

perturbaciones. Para obtener mayor información sobre el controlador se muestra también la respuesta del mismo sistema que en el campo del tiempo, en el campo de la frecuencia.

Mediante el botón *FILTRO* implementado se puede desactivar o activar el filtro *butterworth*, con el activado el sistema con el controlador tiene un margen de ganancia de $-17{,}5$ dB para una frecuencia de 1,26 rad/seg y un margen de fase de 50,1 grados para una frecuencia de 6,38 rad/seg.

En el panel de *Test de control* se seleccionó el boton *A1* para probar el controlado (Ecuación 3.4) con la plataforma mediante el *SIMULINK* (Figura 2–10). Al excitar el sistema con una onda escalonada se obtuvo la respuesta mostrada en la Figura 3–6(a) y al excitarlo con una señal sinusoidal el sistema se comporto como se muestra en la Figura 3–6(b). Las respuestas mostradas nos brindan un correcto desempeño del control sintetizado.



(a) Señal escalón (b) Señal sinusoidal

Figura 3–6: Respuesta del sistema controlado.

Terminado se puede proceder a guardar los datos del trabajo realizado con la interfaz **Control** (Figura 3–5) mediante el botón *GUARDAR*, entre estos datos están: el modelo reducido, el controlador, el filtro, etc. Los datos pueden ser guardados mediante un fichero .mat o .txt.

3.4. Consideraciones finales del capítulo.

Con la validación de la interfaz diseñada se arriban a las siguientes conclusiones:

⋆ Con una PRBS de 0,5 Hz y 70 mm de amplitud centrada a la mitad del cilindro y un controlador proporcional de ganancia 0,019, la respuesta del sistema se nota adecuadamente excitada.

⋆ A pesar de la conocida dinámica no lineal y acoplada del sistema, un modelo ARMAX de tercer orden queda validado con un FIT del 67 % y el resto de los parámetros en los rangos correctos.

⋆ Un controlador con sintetizado para un segundo de tiempo de establecimiento con un filtro de *butterworth* que corta a los 30 Hz, garantiza un margen de ganancia de 17 dB y un margen de face de 50 grados y en la práctica la respuesta cumple con las especificaciones establecidas.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

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

⋆ El *MATLAB*, por su *toolbox* de identificación y su potencialidad para comunicarse en tiempo real con la plataforma, demostró ser una plataforma adecuada para el desarrollo de la interfaz requerida.

⋆ La manipulación de la interfaz desarrollada no requiere el conocimiento profundo de las funciones del *toolbox* de identificación ni del trabajo con *SIMULINK* en tiempo real dado que todo esto es transparente para el usuario quien apenas necesita saber cosas básicas de *MATLAB*. Por esta razón la interfaz desarrollada pudiera emplearse sin mayores dificultades por el personal de SIMPRO.

⋆ El proceso completo de identificación, síntesis y prueba del controlador ha sido realizado en el simulador real con excelentes resultados quedando demostrada su validez.

Recomendaciones

Como recomendaciones se dan las siguientes:

⋆ Para la terminación completa de esta primera versión de la interfaz, debe acompañarse el mismo de un manual de usuario digital que puede estar basado en las ideas expresadas en este documento. Esta ayuda debe ser implementada y agregada a la interfaz.

⋆ Se puede continuar trabajando en la automatización del proceso de identificación de manera que el usuario necesite menos conocimientos del mismo y solo necesite intervenir si el sistema se hace inestable.

⋆ La interfaz, tal y como está, ya puede ser empleado para la identificación y control de los simuladores existentes por lo que se recomienda generalizar su empleo para esta tarea.

⋆ La interfaz pudiera expandirse a tres articulaciones y emplearse en el proceso de identificación y control de las nuevas plataformas de tres GDL.

REFERENCIAS BIBLIOGRÁFICAS

Aracil, Rafael; Saltarén, Roque; Reinoso Oscar (2003). Parallel robots for autonomous climbing along tubular structures. *Elsevier Journal, Robotics and Autonomous Systems* **42**(2), 125–134.

Aracil, Rafael; Saltarén, Roque; Sabater José M; Reinoso Oscar (2006). Robots paralelos: Máquinas con un pasado para una robótica del futuro. *Revista Iberoamericana de Automática e Informática Industrial* **3**, 16–28.

Barragán, Diego Orlando (2007). *MANUAL DE INTERFAZ GRÁFICA DE USUARIO EN MATLAB*.

Carballo, Carlos Raul (2011). Interfaz gráfica para la identificación para la plataforma de 2 grados de libertad. PhD thesis. Universidad Central "Marta Abreu"de Las Villas.

Carducci, G; Foglia, M.; Gentile A.; Giannoccaro N. I.; Messina A. (2004). Pneumatic robotic arm controlled by on-off valves for automatic harvesting based on vision localization. In: *ICIT '04. IEEE International Conference on Industrial Technology*. Vol. 2. pp. 1017–1022.

Chen, Chih-Keng; Hwang, James (2004). terative learning control for position tracking of a pneumatic actuated x-y table. In: *IEEE International Conference on Control Applications*. Vol. 1. pp. 388–393.

Coplien, J.O. (1995). Pattern languages of program design.

Entenza, Pablo José (2009). Diseño de una estructura paralela en ADAMS y su enlace con Simulink. PhD thesis. Universidad Central "Marta Abreu"de Las Villas.

Hernández, Luis; Rubio, Ernesto; Aracil Rafael; Saltarén Roque; Rosell Karel (2004). Identificación y servo control robusto para cilindro actuador neumático. In: *Informática-2004*. La Habana, Cuba.

Instrument, National (2010). Características de análisis y generación de reportes en labview 2010.

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

Kumar, S. (2010). Exploracion de nuevas caracteristicas de c++ y mfc en visual studio 2010.. *MSDN Magazine*.

Ljung, Lennart (1999). *System Identification. Theory for the user*. second edition ed.. Prentice Hall.

López, Y. R. (2010). Modelo Cinemático para Robot Paralelo de tres grados de libertad. PhD thesis. Universidad Central "Marta Abreu"de Las Villas.

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

Murthy, Sunil (2009). *Simuladores Moog*. repas, robert ed.. Moog Industrial Group. East Aurora, N.Y.

Pearce, Michael (2005). Is there an alternative to fluid power?. *IEEE Journal of Computing & Control Engineering* **16**(2), 8–11.

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

Quintana, Raúl Moreno (2000). Plataforma para simuladores. PhD thesis. Instituto Superior Politécnico "José Antonio Echevarría".

Rubio, Angel E.; Hernández, Luis; Alberto Jorge (2011). Identificación y control de la plataforma electro-neumática de tres grados de libertad.

Rubio, Ernesto (2009). Modelación, identificación y control de actuadores electroneumáticos para aplicaciones industriales. PhD thesis. Universidad Central "Marta Abreu"de Las Villas.

Velazco, S. E. D. (2007). Modelo Cinemático y Dinámico para Plataforma de dos grados de libertad. PhD thesis. Universidad Central "Marta Abreu"de Las Villas.

ANEXOS

ANEXOS A
IMPLEMENTACIÓN DE LA INTERFAZ IDENTIFICACIÓN

A.1. Botón MOSTRAR.

Este botón muestra la señal de excitación seleccionada por el usuario (Tren de pulsos o PRBS).

```
if get(handles.RBprbs)
    [t, u] = prbs( ampl, tEst, tExp, frecMax); LLAMA A LA FUNCIÓN PRBS
else
    [t, u] = trendp( ampl, tEst, tExp, frecMax); LLAMA A LA FUNCIÓN TRENDP
end

axes(handles.axes2)
plot(t,u+PO);
```

A.1.1. Señal "PRBS".

```
u = idinput(tt/tm,'PRBS',[0 beta],[-ampl/2 ampl/2])*-1;
u = [zeros(ts/tm,1); u];
t = (0:length(u)-1)'*tm;
```

A.1.2. Señal "Tren de pulsos".

```
yp = [ones(ceil(1/(2*F*T)),1);-ones(ceil(1/(2*F*T)),1)];
```

```
m = size(yp);
mind = (1:m)';
mind = mind(:,ones(1,ceil(Tx*F)));
y = yp(mind,1);

y = [zeros(ceil(Te/T),1);y];

u = y*A/2;
t = (0:length(u)-1)'*T;
```

A.2. Botón SIM-RT.

```
% SE LE ASIGNA pf AL FICHERO DEL SIMULINK
pf = 'mdlrt';
open_system(pf);

% SIMULINK
set_param( pf,'Stop time',num2str(Tsim))
set_param([pf '/Selec'],'Value', num2str(artt));
set_param([pf '/Constant'],'Value', num2str(PO));
set_param([pf '/Constant1'],'Value', num2str(PO));
if artt,
    set_param([pf '/Gain1'],'Gain', num2str(gain));
    set_param([pf '/Gain'],'Gain', num2str(0.01));
else
    set_param([pf '/Gain1'],'Gain', num2str(0.01));
    set_param([pf '/Gain'],'Gain', num2str(gain));
end
% PREVIAMENTE GARANTIZAR QUE EL SIMULINK COMPILE
```

```
% Y SALVARLO YA EN MODO EXTERNO
make_rtw;


set_param(gcs,'SimulationCommand','connect');
set_param(gcs,'SimulationCommand','start');
```

ANEXOS B
IMPLEMENTACIÓN DE LA INTERFAZ ESTIMACIÓN

B.1. Botón PREPROCESAR.

```
% FILTRO BUTTERWORLD
wn = fc*T;
[B,A] = butter (n,wn);


% FILTRADO DE LA SEÑAL (posf)
posf = filter (B,A,pos);
if get(handles.rtendencia,'Value')
    % TENDENCIA
    spt = detrend(sp(1+round(ti/T):round(tf/T)),1);
    post = detrend(pos(1+round(ti/T):round(tf/T)),1);
    posft = detrend(posf(1+round(ti/T):round(tf/T)),1);
else
    % MEDIA
    spt = detrend(sp(1+round(ti/T):round(tf/T)),'constant');
    post = detrend(pos(1+round(ti/T):round(tf/T)),'constant');
    posft = detrend(posf(1+round(ti/T):round(tf/T)),'constant');
end
% MOSTRAR SOLAMENTE LOS SEGMENTOS SELECCIONADOS
axes(handles.axes1)
plot(Tv(1+round(ti/T):round(tf/T)),[spt post posft])
```

B.2. Botón PREPROCESAR.

```
% OBTENER EL MÉTODO DESEADO DEL POPUPMENU
método = get(handles.popupmenu_metodos,'Value');


% %     A(q)y(t) = [B(q)/F(q)] u(t-nk) +  [C(q)/D(q)]e(t)
% %     nb=nc=N  && nf=nd=na=D
% %     M = armax(Z,[na nb nc nk])  %% mod = nb/na
% %     M = arx(Z,[na nb nk])       %% mod = nb/na
% %     M = BJ(Z,[nb nc nd nf nk])  %% mod = nb/nf
% %     M = OE(Z,[nb nf nk])        %% mod = nb/nf


switch método
case    1
    mod = armax(data,[D N N 1]);
    [a,b,c,d,f,da,db,dc,dd,df] = polydata(mod);
    muestracoef( b, db, a, da, handles );
    dens = a;
case    2
    mod = arx(data,[D N 1]);
    [a,b,c,d,f,da,db,dc,dd,df] = polydata(mod);
    muestracoef( b, db, a, da, handles );
    dens = a;
case    3
    mod = bj(data,[N N D D 1],'Focus','Sim');
    [a,b,c,d,f,da,db,dc,dd,df] = polydata(mod);
    muestracoef( b, db, f, df, handles );
    dens = f;
case    4
```

```
        mod = oe(data,[N D 1]);
        [a,b,c,d,f,da,db,dc,dd,df] = polydata(mod);
        muestracoef( b, db, f, df, handles );
        dens = f;
end
    mod.Ts = T; % AÑADIRLE EL TIEMPO DE MUESTREO AL MODELO OBTENIDO
    nums = b;

    [YH,FIT] = compare( data,mod );

    handles.YH = YH;

    axes(handles.axes2)
    plot(Tv(1+round(ti/T):round(tf/T)),[YH{1}.OutputData posft])

    axes(handles.axes3)
    residmod(mod,data);

    set(handles.fit,'String',FIT); %MOSTRAR EL FIT

    % SISTEMA DISCRETO
    SYSD = TF(nums,dens,T);
    % SISTEMA CONTINUO
    SYSC = D2C(SYSD,'tustin');
```

B.2.1. Despeje del sistema continuo en lazo abierto.

```
    % TF CONTINUO EN LAZO CERRADO
    [ncl,dcl] = tfdata(SYSC,'v');
```

```
% DESPEJE DE LA TF EN LAZO ABIERTO

nol = ncl/Kp;

dol = dcl-ncl;

nol = nol/dol(1);

dol = dol/dol(1);


% TF EN LAZO ABIERTO

modol = TF(nol,dol);
```

ANEXOS C
IMPLEMENTACIÓN DE LA INTERFAZ CONTROL

C.1. Síntesis del controlador.

C.1.1. Reducción del modelo.

```
[modb,g] = balreal(modol);


if length(g)>3,
    modr1 = modred(modb,4:length(g),'MatchDC');
else
    modr1 = modb;
end


[Z,P,K] = zpkdata(modr1,'v');
```

C.1.2. Modelo reducido

```
% OBTENER LOS COEFICIENTES NECESARIOS DEL MODELO ESTIMADO
[n,d] = tfdata(modr3,'v');


km = real(n(4)/d(3));
a1m = d(2);
a0m = d(3);
```

```
% MODELO REDUCIDO modr
modr = tf(km*a0m,[1 a1m a0m 0]);
```

C.1.3. Cálculo del controlador.

```
% DECLARAR ESTOS VALORES SIMBÓLICOS
syms km a0m a1m
syms wa f w
syms s kp ki k0 k1

% CONFORMAR EL SISTEMA DE ECUACIONES
eq1 = k1 - (2*wa+2*f*w);
eq2 = k0 - (wa^2+4*wa*f*w+w^2);
eq3 = km*a0m*kp - (2*wa^2*f*w+2*wa*w^2);
eq4 = km*a0m*kp*ki - wa^2*w^2;

% EVALUAR LAS ECUACIONES
eq1e = eval(eq1);
eq2e = eval(eq2);
eq3e = eval(eq3);
eq4e = eval(eq4);

% SOLUCIÓN DE LAS ECUACIONES
S = solve(eq1e,eq2e,eq3e,eq4e,'kp,ki,k0,k1');

i  = 1;
kp = real(eval(S.kp(i)));
ki = real(eval(S.ki(i)));
k0 = (eval(S.k0(i)));
```

```
k1 = (eval(S.k1(i)));


% CONTROLADOR
Cn = tf(conv(kp*[1 ki],[1 a1m a0m]),[1 k1 k0 0]);
```