Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Programa para obtención y procesamiento de datos de identificación desde el MatLab.

Autor: Eduardo Lara Martínez.

Tutores: Msc. Iván Iglesias Navarro.

Dr. Francisco Herrera Fernández.

Santa Clara

2014

"Año 56 del Triunfo de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Programa para obtención y procesamiento de datos de identificación desde el MatLab.

Autor: Eduardo Lara Martínez.

e-mail: elara@uclv.edu.cu

Tutores: Msc. Iván Iglesias Navarro.

e-mail: iglesias@uclv.edu.cu

Dr. Francisco Herrera Fernández.

e-mail: herrera@uclv.edu.cu

Consultante: Javier Alejandro Pérez Pineda

e-mail:javi23pp@yahoo.es

Santa Clara

2014

“Año 56 del Triunfo 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

"Nuestra recompensa se encuentra en el esfuerzo, y no en el resultado. Un esfuerzo total es una victoria completa".

Mohandas K. Gandhi.

DEDICATORIA

A mi familia, en especial a mi madre y hermano por toda la comprensión y apoyo que siempre me han dado.

AGRADECIMIENTOS

A:

Mi mamá, a mi papá, a mi hermano, a mi abuela, a mi cuñada y a mi sobrino por el ejemplo que me han dado y por el apoyo en la realización de este trabajo.

Eliany por representar tanto para mí, y soportarme en los momentos que más triste estaba, a su familia que estaban seguros de que iba a graduarme y que me apoyaron incondicionalmente.

Mis tutores, que a pesar de tener tantos (Herrera, Javier e Iván),me ayudaron para que pudiera terminar esta investigación.

A los profesores del departamento por estos años de guía.

Mis amigos de aula Eliseo, Yissel, Elizabeth, Ronald, Michel, Sandy, Leyanis, Yonder Sergio, por los momentos que compartimos juntos.

Mis amigos de siempre Yusniel y Luis Emilio, por las historias juntos.

Dailén, Yisliany, Barbará y Aramí por la amistad de siempre.

Mi otra familia el 5 de diciembre que siempre supieron darme consuelo y apoyo, no solo los que están sino también los que ya se fueron, dígase Rafa, Yander, Clara, Rosario, Rosmery, Lisandra que confió en mí para bailar La Candela junto a Daimé, a Regla, Merlyn la blanca, la pitineja, la negra, Daily, Arianna, Diana, Marianny, Any, a Geisy y Laira que siempre me dieron ánimo para que superara mis miedos, a Lander, Angel, Ariel, Alejandro, Julio, Diuvel, a nuestro director Gerardo, y por último y no menos importantes a los otros que componen los 5 elementos Yasser, Karell y Youry ,gracias de verdad a todos.

RESUMEN

Actualmente el proceso de Identificación de Sistemas constituye un elemento fundamental en la obtención de modelos matemáticos, empleados en el campo de la ingeniería. Una de las vías para optimizar este proceso sería la utilización de un Sistema Multi-Agente que sustituya la función del ingeniero al seleccionar el mejor modelo entre los identificados. Con la implementación de este sistema surge la necesidad de crear una base de datos en la que se apoyaría en el proceso de toma de decisiones a la hora de seleccionar el modelo válido.

Según tales demandas, la presente investigación propone una función para almacenar los datos provenientes del proceso de identificación no paramétrica en MATLAB en una base de datos. Para ello, se explota la capacidad del Toolbox de Identificación de Sistemas de MATLAB de proporcionar los datos que más información brindan acerca del comportamiento de las diferentes estructuras. Asimismo se utilizó el Toolbox de Base de Datos de MATLAB, el cual permite establecer una comunicación con el sistema gestor MySQL, en el cual se crea la base de datos relacional donde se va a almacenar dicha información.

Se obtiene como resultado la función SaveIntoDbNp que permite crear la base de datos relacional DATA_BASE en MySQL, así como obtener la información que brinda MATLAB de los modelos estimados seleccionados por el usuario y exportarla a dicha base de datos. La cual posee una estructura simple que posibilita el eficiente almacenamiento y la fácil recuperación de los datos para que sean utilizados en análisis posteriores.

Palabras Claves—IDENTIFICACIÓN DE SISTEMAS, MODELOS, BASE DE DATOS.

CONTENIDO

INTRODUCCIÓN

A través de los años, la identificación de sistemas se ha convertido en una herramienta importante en la ingeniería y otras áreas tan diversas como medicina y economía, entre otras, que requieren de modelos que posibiliten el análisis, la simulación y el diseño e implementación de estrategias de control.

Solucionar problemas en la identificación de procesos constituye un área de investigación multidisciplinaria, que tiene muchas importantes aplicaciones.

Cuando interactuamos con el entorno, intuitivamente aprendemos a controlar nuestras acciones prediciendo su efecto. Esas predicciones están basadas en un modelo innato adaptado a la realidad, usando nuestras experiencias pasadas.

Durante un largo período los modelos creados por el hombre para explicar sus observaciones del universo eran cualitativos; pero en los últimos siglos se han complementado con modelos cuantitativos basados en matemática avanzada.

Con la insaciable demanda y el desarrollo de la ciencia y la tecnología, los instrumentos para la obtención de modelos se han perfeccionado y son extensivamente usados en todos los campos, incluyendo las áreas no técnicas como la economía, ecología y la biología.

Saber el comportamiento de las variables en las plantas de una industria es una necesidad, debido a que son procesos complejos que necesitan tener modelos matemáticos, trabajando en tiempo real, con el objetivo de describir su comportamiento.

En la actualidad, es habitual el uso de herramientas de modelado computacional en la descripción de estos procesos. Especialmente un paso concreto en el desarrollo de sistemas de estimación es el empleo del toolbox de identificación de sistemas (Systems Identification) del MatLab, que contiene funciones para la estimación de modelos de sistemas dinámicos,(Ljung, 2007)

La importancia de la identificación de sistemas radica, en que puede ser usado un método de obtención experimental de determinado modelo. El mismo, para los fines deseados, debe reproducir con suficiente exactitud las características dinámicas del proceso objeto de

estudio. Es importante lograr una buena identificación que proporcione modelos de orden reducido más fáciles de utilizar,(Ljung, 1999).

La identificación es un proceso iterativo donde el hombre tiene la tarea de planificar el experimento, seleccionar la estructura y el método de ajuste adecuado, para validar el modelo que cumpla las expectativas. Es imperativo que este proceso sea lo más confiable, óptimo y exacto posible, por lo que la mejor opción sería que lo realizara un programa. A partir de este supuesto, se establece como primicia en la investigación utilizar los datos que brinda el toolbox de MatLab para que sean usados por dicho programa.

Sin embargo es necesario almacenar estos datos en una base de datos y que así sea más fácil el acceso a ellos para un análisis posterior. Surge entonces la necesidad de desarrollar una aplicación encargada de obtener los datos resultantes del proceso de identificación no paramétrica; estableciendo una comunicación entre MatLab y la base datos donde serán almacenados. Lo cual representa el principal aporte de la investigación.

Antecedentes:

Con los cambios en la tecnología de base de datos y las mejoras en el rendimiento y el almacenamiento en disco, las reglas han cambiado y ahora tiene sentido comercial utilizar bases de datos para almacenar y gestionar todos los activos de una empresa o industria. Las siguientes propiedades son las fortalezas que una base de datos puede ofrecer, comparado con el sistema de almacenamiento de archivos tradicional: administración, seguridad, backup / recuperación, extensibilidad, flexibilidad.

Existen varias vías para el almacenamiento en una base datos, aprovechando la capacidad del Database Toolbox de MatLab de conectar con una base de datos, en la actualidad se ha utilizado en numerosas investigaciones a nivel internacional.

Así mismo en 2009 en la Universidad Pontificia Bolivariana, Bolivia se desarrolló la tesis de grado “OBTENCIÓN DEL MODELO NO PARAMÉTRICO DE UN SISTEMA POR EL MÉTODO DE IDENTIFICACIÓN DE RESPUESTA EN FRECUENCIA”. Para ello, decidieron utilizar *Sybase Adaptive Server Enterprise* como base de datos principales Mathworks MATLAB y para manipular datos matemáticos.

En el ámbito nacional aparecen pocas investigaciones relacionadas con el tema. Al respecto el Instituto de Ocenología presentó la investigación: “Herramienta computacional para el cálculo no paramétrico”.

El mismo está basado en un análisis estadístico de una base de datos en tiempo real. Para el procesamiento de la información se cuenta con el software (CONEM) que tiene un conjunto de funciones ejecutables en MatLab 7.0, este software realiza los cálculos por separado en ambiente Excel y este a la vez a ambiente MatLab.

También se puede hacer referencia al Trabajo de Diploma realizado por Javier Alejandro Pérez Pineda que presentó la investigación: “Función para la obtención de datos de identificación desde el MATLAB”. Este trabajo está basado en la creación de una función capaz de identificar el modelo utilizado a partir de datos experimentales apoyándose en el facilitador matemático MatLab, para ello utiliza una base de datos en MySQLWorkbench.

Situación problémica:

En la actualidad no existe función encargada de almacenar en base de datos y analizar los datos resultantes de la identificación no paramétricas usando la herramienta Ident (toolbox) de MatLab.

Problema científico:

¿Cómo almacenar y analizar a través del uso de funciones los datos provenientes de la identificación no paramétrica del Ident (toolbox) de MATLAB en un formato de base de datos?

Objetivo general:

Proponer funciones para la comunicación MATLAB-base de datos que sean capaces de almacenar y procesar datos resultantes de diferentes procedimientos de identificación realizados en MATLAB, posibilitando con ello efectuar análisis de calidad de tales procedimientos para casos particulares.

Objetivos específicos:

Definir cuáles son las variables a tener encuentra para el análisis de los modelos identificados no paramétricos.

Definir cuál software utilizar para la confección de la base de datos.

Diseñar una base de datos que sea capaz de almacenar los datos provenientes del proceso de identificación no paramétrica.

Diseñar y programar las funciones "**SaveIntoDbNp**" y "**Qidem**" para salvar los datos en una base de datos y realizar el análisis de la calidad del procedimiento de identificación, respectivamente.

Interrogantes Científicas:

¿Cómo acceder a los datos obtenidos durante el proceso de identificación no paramétricos con el Ídem de MatLab?

¿Cómo almacenar estos datos en la base de datos?

¿Qué formato de base de datos será la más adecuada?

¿Cómo programas las funciones encargadas del almacenamiento y análisis de los datos de la identificación no paramétrica?

Tareas de investigación:

- Análisis bibliográfico sobre identificación no paramétrica con Ident de MatLab.
- Análisis bibliográfico sobre comunicación entre aplicaciones.
- Análisis bibliográfico sobre base de datos más utilizados, en la comunicación con MATLAB.
- Estudio de MatLab y su herramienta Ident.
- Programación de la aplicación.
- Pruebas y Análisis de los resultados.
- Informe.

Resultados:

Con la realización de este proyecto se pretende tener como resultado dos funciones, encargadas: una de obtener los datos resultantes del proceso de identificación no paramétrica y almacenarlos en una base de datos y otra que se encargar de seleccionar de todos los modelos identificados cual será el mejor para cada uno de los índices de bondad de ajustes que se definan. Se contará con la base de datos (SQL Server) pertinente para establecer una comunicación con MatLab, en la cual serán almacenados los datos, que serán usados en análisis posteriores.

Impacto:

La investigación contribuirá a responder problemáticas actuales relacionadas con la implementación de software en la obtención experimental de modelos. Por lo que constituirá un aporte para la comunidad científica y técnica.

Con el diseño de la aplicación que supone la investigación se contribuirá al desarrollo de posteriores investigaciones que persiguen la optimización de la identificación de sistemas.

Aplicabilidad:

Los resultados de la investigación poseen una aplicación práctica y metodológica de los procedimientos de identificación, en tanto constituye un paso más en la optimización de estos procedimientos de identificación.

Viabilidad:

Se puede afirmar que se cuenta con los recursos humanos y materiales necesarios. Así mismo el proyecto se ajusta a demandas actuales y se ubica racionalmente en el tiempo y en el contexto social. Es importante destacar que se emplea un gestor de base de datos libres (o sea software libre).

El informe de la investigación está estructurado de la siguiente manera: La presente introducción a la que siguen, en el mismo orden, capitulario, conclusiones, recomendaciones, referencias bibliográficas y anexos.

Los temas abordados en cada uno de los capítulos son:

Capítulo 1: Las bases de datos, su uso para apoyar el proceso de identificación de sistemas. Se realiza una revisión bibliográfica acerca del tema concluyendo que la base de datos a usar en la investigación es el MySQL por las propiedades que esta posee.

Capítulo 2: Funciones **"SaveIntoDbNp." Y "Qidem",** para el almacenamiento y análisis de los datos obtenidos en la identificación "No paramétrica" con Ident del MatLab. Haciendo uso de los datos a obtener en la identificación se crea la función saveintodbnp para que salve los datos de este proceso y luego haciendo procesar estos datos por la función Qidem obtenemos el mejor modelo a partir del cálculo de los índices de bondad expuestos en la investigación.

Capítulo 3: Prueba y Análisis de los resultados.

Capítulo 1. Las bases de datos, su uso para apoyar el proceso de identificación de sistemas.

Una base de datos proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado. Se convierte más útil a medida que la cantidad de datos almacenados crece.

Una base de datos puede ser local, es decir que puede utilizarla sólo un usuario en un equipo, o puede ser distribuida, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red.

La principal ventaja de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo,(Silberschatz, 2002).

1.1 ¿Qué es una base de datos?

Una base de datos (cuya abreviatura es *BD*) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Diferentes programas y diferentes usuarios deben poder utilizar estos datos. Por lo tanto, el concepto de base de datos generalmente está relacionado con el de red ya que se debe poder compartir esta información. De allí el término **base**. “Sistema de información” es el término general utilizado para la estructura global que incluye todos los mecanismos para compartir datos que se han instalado,(Sánchez, 2004).

1.1.1 Caracteristicas de las bases de datos.

Una base de datos contiene entidades de información que están relacionadas vía organización y asociación. La arquitecturalógica de una base de datos se define mediante un esquema que representa las definiciones de las relaciones entre las entidades de información. La arquitectura física de una base de datos depende de la configuración del hardware residente. Sin embargo, tanto el esquema (descripción lógica) como la organización (descripción física) deben adecuarse para satisfacer los requerimientos

Las bases de datos, su uso para apoyar el proceso de identificación se sistemas. funcionales y de comportamiento para el acceso al análisis y creación de informes,(Sánchez, 2004).

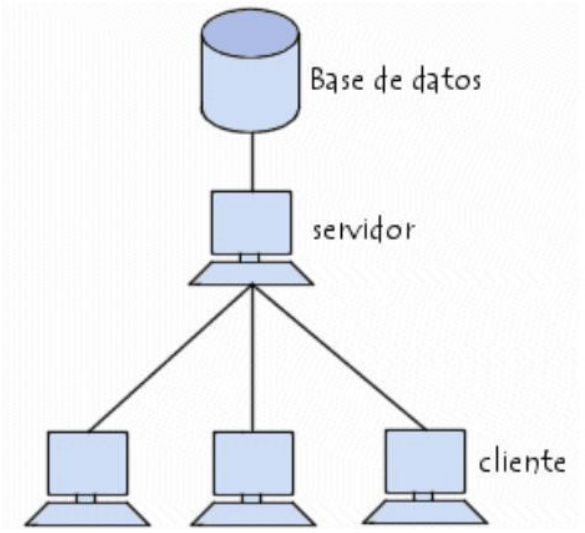


Figura 1.1 Estructura de una base de datos.

1.1.2 Componentes de una Base de Datos

- Hardware: constituido por dispositivo de almacenamiento como discos, tambores, cintas, etc.
- Software: que es el DBMS o Sistema Administrador de Base de Datos.
- Datos: los cuales están almacenados de acuerdo a la estructura externa y van a ser procesados para convertirse en información.

1.1.3 Conceptos Básicos de Base de datos

- Archivo: son conjuntos de registros.
- Registros: son conjuntos de campos.
- Campos: es la mínima unidad de referencia.

1.1.4 Niveles de Abstracción en Base de datos

- Externo: esa es la visión del usuario final, se ve cómo se maneja los datos ya convertidos en información. Es aquel en el que se presenta al usuario final y que puede combinaciones o relaciones entre los datos que conforman a la base de datos global. Puede definirse como la forma en el que el usuario aprecia la información y sus relaciones.
- Conceptual: se ve como está estructurado la Base Datos, equipos de campo tiene como están estructura los registros. Se definen las estructuras lógicas de almacenamiento y las relaciones que se darán entre ellas.

Ejemplos comunes de este nivel son el diseño de los registros y las ligas que permitirán la conexión entre registros de un mismo archivo, de archivos distintos incluso, de ligas hacia archivos.

- Interno: se ve como se almacena los datos físicamente. Se determinan las características de almacenamiento en el medio secundario. Los diseñadores de este nivel poseen un amplio dominio de cuestiones técnicas y de manejo de hardware. Muchas veces se opta por mantener el nivel físico proporcionado por el sistema operativo para facilitar y agilizar el desarrollo.

1.2 Administración de bases de datos

Para el desarrollo de sistemas de bases de datos se requiere mucho más que únicamente la selección de un modelo lógico de base de datos. La bases de datos es una disciplina organizacional, un método, más que una herramienta o una tecnología. Requiere de un cambio conceptual y organizacional. Rápidamente surgió la necesidad de contar con un sistema de administración para controlar tanto los datos como los usuarios. La administración de bases de datos se realiza con un sistema llamado **DBMS** (DatabasemanagementSystem [Sistema de administración de bases de datos]). El DBMS es

Las bases de datos, su uso para apoyar el proceso de identificación se sistemas. un conjunto de servicios (aplicaciones de software) para administrar bases de datos, que permite:

- un fácil acceso a los datos
- el acceso a la información por parte de múltiples usuarios
- la manipulación de los datos encontrados en la base de datos (insertar, eliminar, editar),(Cuadra, 2000).

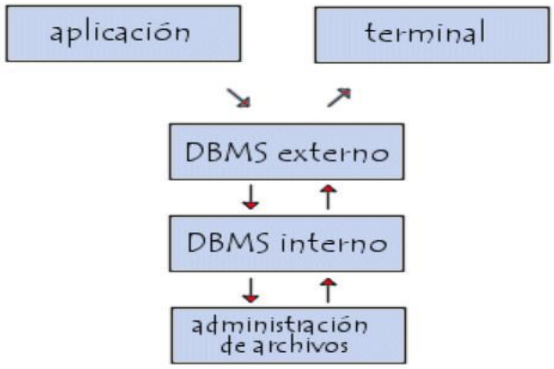


Figura 1.2 Administración de las base de datos.

El DBMS puede dividirse en tres subsistemas:

- El sistema de administración de archivos: para almacenar información en un medio físico.
- El DBMS interno: para ubicar la información en orden.
- El DBMS externo: representa la interfaz del usuario.

Los principales sistemas de administración de bases de datos son:

- BorlandParadox
- Filemaker
- IBM DB2
- Ingres

- Interbase
- Microsoft SQL server
- Microsoft Access
- Microsoft FoxPro
- Oracle
- Sybase
- MySQL
- PostgreSQL
- mSQL
- SQL Server 11

1.3 Requerimientos de las bases de datos:

El análisis de requerimientos para una base de datos incorpora las mismas tareas que el análisis de requerimientos del software. Es necesario un contacto estrecho con el cliente; es esencial la identificación de las funciones e interfaces; se requiere la especificación del flujo, estructura y asociatividad de la información y debe desarrollarse un documento formal de los requerimientos,(Cuadra, 2000).

Elementos claves de organización en un ambiente de Bases de Datos:

- Sistema de administración de base de datos
- Administración de información
- Tecnología de administración de base de datos
- Usuarios
- Planeación de información y tecnología de modelaje

1.4 Ventajas en el uso de bases de datos:

La utilización de bases de datos como plataforma para el desarrollo de Sistemas de Aplicación en las Organizaciones se ha incrementado notablemente en los últimos años, se

debe a las ventajas que ofrece su utilización, algunas de las cuales se comentarán a continuación:

- Globalización de la información: permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.
- Eliminación de información inconsistente: si existen dos o más archivos con la misma información, los cambios que se hagan a éstos deberán hacerse a todas las copias del archivo de facturas.
- Permite compartir información.
- Permite mantener la integridad en la información: la integridad de la información es una de sus cualidades altamente deseable y tiene por objetivo que sólo se almacena la información correcta.
- Independencia de datos: el concepto de independencia de datos es quizás el que más ha ayudado a la rápida proliferación del desarrollo de Sistemas de Bases de Datos. La independencia de datos implica un divorcio entre programas y datos.

1.5 Ambiente moderno de base de datos:

La tecnología de las bases de datos puede eliminar de un tajo muchos problemas creados por la organización tradicional de archivos. Una definición más rigurosa de bases de datos dice que es una colección de datos organizada para dar servicios eficientemente a muchas aplicaciones al centralizar los datos y minimizar aquellos que son redundantes. En vez de separar los datos en archivos por separados para cada aplicación, los datos son almacenados físicamente para aparecer a los usuarios como almacenados en una sola ubicación: una sola base de datos sirve a muchas aplicaciones. Por ejemplo, en vez de que una corporación almacene los datos de personal en sistemas de información separados y archivos separados para personal, nominas y prestaciones, la corporación podría crear una sola base de datos para Recursos Humanos,(Sánchez, 2004).

1.6 Diseño de una base de datos

Existen distintos modos de organizar la información y representar las relaciones entre los datos en una base de datos. Los Sistemas administradores de bases de datos convencionales usan uno de los tres modelos lógicos de bases de datos para hacer seguimiento de las entidades, atributos y relaciones. Los tres modelos lógicos principalmente de bases de datos son el jerárquico, de redes y el relacional. Cada modelo lógico tiene ciertas ventajas de procesamiento y también ciertas ventajas de negocios,(Sánchez, 2004).

1.6.1 Modelo de jerárquico de datos:

Una clase de modelo lógico de bases de datos que tiene una estructura arborescente. Un registro subdivide en segmentos que se interconectan en relaciones padre e hijo y muchos más. Los primeros sistemas administradores de bases de datos eran jerárquicos. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos.

1.6.2 Modelos de datos en la red:

Es una variación del modelo de datos jerárquico. De hecho las bases de datos pueden traducirse de jerárquicas a en redes y viceversa con el objeto de optimizar la velocidad y la conveniencia del procesamiento. Mientras que las estructuras jerárquicas describen relaciones de muchos a muchos.

1.6.3 Modelo relacional de datos:

Es el más reciente de estos modelos, supera algunas de las limitaciones de los otros dos anteriores. El modelo relacional de datos representa todos los datos en la base de datos como sencillas tablas de dos dimensiones llamadas relaciones. Las tablas son semejantes a los archivos planos, pero la información en más de un archivo puede ser fácilmente extraída y combinada.

1.7 Creación de una base de datos

Para crear una base se deben realizar dos ejercicios de diseño: un diseño lógico y uno físico. El diseño lógico de una base de datos es un modelo abstracto de la base de datos desde una perspectiva de negocios, mientras que el diseño físico muestra como la base de datos se ordena en realidad en los dispositivos de almacenamiento de acceso directo. El diseño físico de la base de datos es llevado a cabo por los especialistas en bases de datos, mientras que el diseño lógico requiere de una descripción detallada de las necesidades de información del negocio de los negocios actuales usuarios finales de la base. Idealmente, el diseño de la base será una parte del esfuerzo global de la planeación de datos a nivel institucional. El diseño lógico de la base de datos describe como los elementos en la base de datos han de quedar agrupados. El proceso de diseño identifica las relaciones entre los elementos de datos y la manera más eficiente de agruparlos para cumplir con los requerimientos de información. El proceso también identifica elementos redundantes y los agrupamientos de los elementos de datos que se requieren para programas de aplicaciones específicos. Los grupos de datos son organizados, refinados y agilizados hasta que una imagen lógica general de las relaciones entre todos los elementos en la base de datos surja,(Silberschatz, 2002).

1.7.1 Bases de datos documentales:

Son las derivadas de la necesidad de disponer de toda la información en el puesto de trabajo y de minimizar los tiempos del acceso a aquellas informaciones que, si bien se utilizan con frecuencia, no están estructuradas convenientemente. Esto se debe a que la procedencia de la información es muy variada (informes, notas diversas, periódicos, revistas, muchos más.

1.7.2 Bases de datos distribuidas:

Es aquella que se almacena en más de un lugar físico. Partes de la base de datos se almacena físicamente en un lugar y otras partes se almacenan y mantienen en otros lugares. Existen dos maneras de distribuir una base de datos. La base de datos central puede ser

particionada de manera que cada procesador remoto tenga los datos necesarios sobre los clientes para servir a su área local. Los cambios en los archivos pueden ser justificados en la base de datos central sobre las bases de lotes, en general por la noche. Otra estrategia también requiere de la actualización de la base central de datos en hojas no laborables. Aun otra posibilidad (una que se emplea en bases de datos grandes) es mantener solo un índice central de nombres y almacenar localmente los registros completos. Los procesamientos distribuidos y las bases de datos distribuidas tienen como beneficios e inconvenientes. Los sistemas distribuidos reducen la vulnerabilidad de un lugar único central y voluminoso. Permiten incremento en la potencia de los sistemas al adquirir mini computadoras que son más pequeñas y baratas. Finalmente incrementan el servicio y la posibilidad de respuesta de los usuarios locales. Los sistemas distribuidos, sin embargo, dependen de la alta calidad de las líneas de telecomunicaciones , las cuales a su vez son vulnerables. Además, las bases de datos locales pueden algunas veces alejarse de las normas y las definiciones de los datos centrales y hacen surgir problemas de seguridad al distribuir ampliamente el acceso a datos de alta sensibilidad.

1.7.3 Bases de datos orientadas a objetos e hipermedia:

Estas son capaces de almacenar tanto procesos como datos. Por este motivo las bases orientadas al objeto deben poder almacenar información no convencional (como imágenes estáticas o en movimiento, colecciones de sonidos, entre otros).Este tipo de bases de datos deriva directamente de la llamada programación orientada a objetos, típica por ejemplo del lenguaje C/C++.

Entre las ventajas de las bases de datos orientadas al objeto destaca la posibilidad de tratar los casos excepcionales, que suelen ser la mayoría en la práctica cotidiana, en lugar de tratar de insertar la realidad en unos patrones rígidos que violentan para hacerla coincidir con los esquemas utilizados. Además, nadie pone en duda que es más cómodo manejar objetos de entorno que no es familiar, que trabaja, por ejemplo, con tablas, esquemas, cuadros, muchos más.

1.8 Sistema de gestión de base de datos apropiado

En la industria moderna uno de los activos más valiosos son los datos y la disposición de ellos. Hacer un mejor uso de los datos garantiza que la industria cumpla sus metas. No obstante, los datos son solo bits y bytes en un sistema de archivos y solo con un DBMS, puedes convertirlos en información seria. Esto significa que es imperativo escoger el sistema gestor de base de datos correcto.

Para lograr una elección formidable se deben tener en cuenta algunos factores como: que sea de fácil uso, su desempeño, seguridad, entre otros. Apoyarse en las tendencias actuales disminuye este trabajo en gran medida.

En este sentido, DBMS como Microsoft Office Excel y Access, Sybase, Oracle y MySQL 5.5 se emplean frecuentemente en proyectos que presentan similitud al presente trabajo.

Cada uno de ellos tiene sus ventajas y desventajas; por ejemplo: Microsoft Office Excel está diseñado para manejar hojas de cálculo y es utilizado normalmente, para tareas financieras y contables. Microsoft Office Access permite crear ficheros de bases de datos relacionales, que pueden ser fácilmente gestionadas por una interfaz gráfica sencilla. Sin embargo, no son suficientemente profesionales y se necesita el paquete de Office.

Por otra parte Sybase y Oracle son DBMS de alto rendimiento, con soporte a grandes volúmenes de datos, pero pertenecen a empresas comerciales, lo cual resulta, en términos de costo, una dificultad.

Finalmente, MySQL es el DBMS más popular a nivel mundial, pues es adoptado mayoritariamente en cuantiosas industrias. Es conocido por su confiabilidad, su fácil uso, flexibilidad, desempeño, y que es un software libre. Conjuntamente, la comunidad que usa MySQL es muy activa, por lo que contar con ayuda y guía es muy efectivo. Es importante destacar que dicho DBMS es soportado por MATLAB y puede establecerse una conexión con él, a través del Toolbox de base de datos de MATLAB. De esta forma, el gestor de base

de datos MySQL es considerado el más apropiado para cumplir los objetivos de la investigación.

1.8.1 Términos básicos

Para comentar sobre estos términos se utiliza como referencia el libro La Biblia de MySQL de 2003.(Gilfillan, 2003)

Los datos son los valores que se almacenan en la base de datos. Por sí solos no tienen mucho significado.

Una base de datos es un conjunto de tablas. Cada tabla se compone de registros (las filas horizontales de la tabla, que también se conocen como tuplas). Cada registro debe ser exclusivo y puede almacenarse en cualquier orden dentro de la tabla. Cada registro se compone de campos (las columnas verticales, que también se conocen como atributos).

Los campos pueden ser de varios tipos. MySQL consta de una gran cantidad de tipos, pero por regla general se pueden clasificar en tres categorías: caracter, numérico y de fecha. El rango de valores permitidos para un campo se conoce como dominio (o especificación del campo). De un campo se dice que contiene un valor nulo cuando no incluye ningún valor. Los campos nulos pueden complicar los cálculos y generar problemas de exactitud en los datos. Por esta razón, muchos campos se configuran de forma que no puedan contener valores nulos. Además, un campo puede ser de incremento automático, estos se asocian con la llave primaria y permiten que MySQL se encargue de la secuenciación del campo. Si se inserta un registro, MySQL agregará una unidad al valor anterior incrementado automáticamente y lo utilizará como valor para el campo de incremento automático utilizado.

Una clave accede a registros especiales de una tabla. Un índice es un mecanismo que mejora el rendimiento de una base de datos. Los índices se suelen confundir con las claves.

Estrictamente hablando, se integran en la estructura física mientras que las claves son parte de la estructura lógica. Sin embargo, estos términos se suelen utilizar indistintamente.

Una relación uno a uno (1: 1) es una relación en la que para cada instancia de la primera tabla de una relación solo existe una instancia en la segunda.

Una relación uno a varios (1: V) es una relación en la que para cada instancia de la primera tabla existen varias instancias en la segunda tabla. Este tipo de relación es muy habitual.

Una relación varios a varios (V: V) es una relación que tiene lugar cuando para cada instancia de la primera tabla existen varias instancias en la segunda, y para cada instancia de la segunda existen varias instancias en la primera.

Una relación obligatoria es una relación en la que para cada instancia de la primera tabla de la relación debe existir una o varias instancias en la segunda.

Una relación opcional es una relación en la que para cada instancia de la primera tabla de una relación pueden existir instancias en la segunda.

Una clave de tabla, como indica el término, desbloquea el acceso a las tablas. Si conocemos la clave, sabremos cómo identificar sus registros así como las relaciones entre las tablas. Una clave candidata es un campo o una combinación de campos que identifiquen un registro de manera exclusiva. No puede contener un valor nulo y su valor debe ser exclusivo. (La existencia de duplicados impide la identificación de un registro exclusivo).

Una clave primaria es una clave candidata que ha sido designada para identificar de forma única los registros de una tabla en la estructura completa de una tabla.

Clave externa: Una relación entre dos tablas se establece asignando un campo común en ellas. Este campo común debe ser la clave primaria de una de estas dos tablas. El cual es denominado como clave externa en la segunda. Las claves externas permiten garantizar lo

que se conoce como integridad referencial[1]. También permiten realizar eliminaciones y actualizaciones en cascada.

El proceso de eliminación recorre las tablas pertinentes, suprimiendo todos los registros procedentes, utilizando una única instrucción, por ejemplo:

Existen dos tablas con una relación entre ellas (1: V), esta relación es posible mediante un campo común que es clave primaria en una y clave externa en la otra. Si se elimina una instancia de la tabla en la cual el campo común es clave primaria, por defecto se eliminan todas las instancias correspondientes en la segunda tabla.

1.8.2 Nivel de acceso a la base de datos en MySQL

Para abordar estos aspectos se utiliza como referencia el libro "MySQL 5.5 Reference Manual" de 2012.(Oracle, 2012)

MySQL cuenta con un sistema de privilegios y contraseñas que proporciona gran seguridad a los datos. Este sistema es muy flexible y seguro, y permite verificación basada en el anfitrión (host). Las contraseñas son seguras porque todo el tráfico de contraseñas está cifrado cuando se conecta con un servidor. Para argumentar estas ideas se abordan los siguientes elementos.

1.9.2-1 El sistema de privilegios de acceso de MySQL

La función primaria del sistema de privilegios de MySQL es autenticar un usuario que se conecta a partir de un anfitrión (host) dado y asociar dicho usuario con privilegios en una base de datos, como SELECT, INSERT, UPDATE y DELETE.

---

[1]*Integridad referencial: Permite borrar un registro que esté "enlazado" por medio de una "relación" entre tablas sin que se cumplan ciertas dependencias, de esta forma no se tienen datos huérfanos que puedan ser inconsistentes con el sistema* (Schwartz et al., 2012).

Funcionalidad adicional incluye la habilidad de tener usuarios anónimos y concederles privilegios para funciones específicas de MySQL, tales como, operaciones administrativas LOAD DATA INFILE.

Internamente, el servidor guarda información de privilegios en las tablas de permisos de la base de datos MySQL (es decir, en la base de datos llamada MySQL). El servidor MySQL lee el contenido de estas tablas en la memoria cuando se inicia y las bases de las decisiones de control de acceso a las copias en memoria de las tablas de permisos.

El sistema de privilegios de MySQL asegura que todos los usuarios pueden realizar solo las operaciones permitidas a los mismos.

Como usuario, cuando se conecta a un servidor MySQL, su identidad se determina mediante el equipo desde el que se conecta y el nombre de usuario que se especifique. Cuando efectúe peticiones tras conectar, las subvenciones del sistema privilegios de acuerdo a su identidad y lo que quieres hacer.

MySQL considera tanto su nombre de host y el nombre de usuario al identificarle. Ya no hay razón por la cual asumir que un nombre de usuario dado pertenece a la misma persona en todos los anfitriones (hosts).

1.9.2-2 Control de acceso de MySQL

Implica dos etapas cuando se ejecuta un programa cliente que se conecta con el servidor:

Etapa 1: El servidor acepta o rechaza la conexión a la base de su identidad y si se puede verificar su identidad proporcionando la clave correcta.

Etapa 2: Asumiendo que se puede conectar, el servidor comprueba cada comando que ejecuta para determinar si tiene privilegios suficientes para llevarla a cabo.

Si se cambian (por el usuario o alguien más) sus privilegios mientras está conectado, los cambios no tienen por qué tener efecto inmediatamente para la próxima sentencia que se emite.

1.9.2-3 MySQL Workbench

Según Oracle (2012) MySQLWorkbench es una herramienta gráfica para trabajar con servidores MySQL (versión 5.1 en adelante) y base de datos. Sus tres principales áreas de funcionalidad son:

- **Desarrollo de SQL:** Permite crear y administrar conexiones a servidores de base de datos. Así como configurar los parámetros de una conexión. Además, provee la capacidad de ejecutar consultas de SQL en las conexiones de base de datos usando el editor SQL built-in.
- **Modelado de datos:** Permite crear modelos de esquemas de base de datos gráficamente, así como ingeniería inversa y directa entre un esquema y una base de datos en vivo y editar todos los aspectos de la base de datos usando el Editor de Tablas. Este editor provee facilidades fáciles de usar para editar: tablas, columnas, índices, inserciones, privilegios, entre otras.
- **Administración de servidores:** Permite crear y administrar instancias de servidores.

1.10 Uso de la bondad de ajuste para el proceso de identificación de sistemas.

Se entiende por *bondad de ajuste*, el grado de acoplamiento que existe entre los datos originales y los valores teóricos que se obtienen de la regresión. Se trata de saber si el modelo que se ha ajustado para relacionar las variables *X* e *Y* es consistente. La medida más comúnmente usada para medir el ajuste de la recta de regresión es el ***coeficiente de correlación lineal (r):***

$$r = \frac{S_{xy}}{S_x S_y} \tag{1}$$

Donde S_{xy}es la covarianza muestral, $S_x$la desviación típica muestral de la variable X yS_y la desviación típica muestral de la variable Y.

Estos datos se determinan mediante las siguientes expresiones:

$$S_{xy} = \frac{(\sum_i (x_i - \bar{x})(y_i - \bar{y}))}{n} = \left(\frac{\sum_i x_i y_i}{n}\right) - \overline{xy} \tag{2}$$

$$S_x = \sqrt{\sum_i \frac{(x_i - \bar{x})^2}{n}} = \left(\sqrt{\frac{(\sum_i x_i^2)}{n - \bar{x}^2}}\right) \tag{3}$$

$$S_y = \sqrt{\sum_i \frac{(y_i - \bar{y})^2}{n}} = \left(\sqrt{\frac{(\sum_i \mathbf{y_i^2})}{\mathbf{n} - \bar{\mathbf{y}}^{\mathbf{2}}}}\right) \tag{4}$$

El coeficiente de correlación lineal es un valor que cumple las condiciones siguientes:

- Toma valores entre −1 y 1.
- Es invariante por transformaciones lineales de las variables X e Y.

- Cuanto más extremo es r (más se acerca a −1 o a 1), significa que mejor se ajusta el modelo.

Un valor cercano a 0, debe interpretarse como que no existe un buen ajuste lineal, pero ello no excluye que existan otros tipos de relaciones funcionales. De hecho, podemos hallar ejemplos de relaciones funcionales exactas, con un coeficiente de correlación 0.
Por otro lado, un coeficiente de correlación muy cercano a 1 o a −1, no debe interpretarse como que existe una relación causa-efecto importante entre las dos variables. La relación podría deberse al efecto de otras variables no incluidas en el estudio.

El cuadrado del valor r multiplicado por 100 se denomina coeficiente de determinación y se interpreta como el porcentaje de variabilidad que explica el modelo.

1.10.1 Coeficiente de determinación

Según (Posada and Noguera, 2007). El coeficiente de determinación mide el porcentaje de variación total en Y debido a las variables que toma el investigador Este valor se obtiene a partir de la *suma de los cuadrados del error (SCE)* y de la *suma de los cuadrados totales (SCT)*, a partir de la ecuación.

$$R^2 = 1 - \frac{SCE}{SCT} \tag{5}$$

La *SCE* corresponde a la suma de los cuadrados de las distancias de los puntos desde la curva de mejor ajuste, en tanto la *SCT* es la suma de los cuadrados de las distancias de los puntos desde una línea horizontal correspondiente a la media de todos los valores de *Y*.

1.10.2 Cuadrado medio del residuo (o del error) CME

Según (Posada and Noguera, 2007)]. *CME* es una medida que agrupa la variabilidad de aquellos factores que no tiene en cuenta el investigador. La varianza de *n* residuales e_i se representa como:

$$CME = \frac{\sum(e_i - \bar{e})^2}{n-K} = \frac{\sum e_i^2}{n-K} = \frac{SCE}{n-K} \quad (6)$$

Donde, $\bar{e}$ es la media de *n* residuales (número de observaciones),*K* es el número de parámetros estimados en el modelo y *SCE* es la suma de cuadrados de las distancias verticales de los puntos desde la curva de regresión.

Toda vez que el *CME* corresponde a la varianza residual, los modelos seleccionados por su mayor capacidad de ajuste son aquellos que expresan el menor valor en este criterio.

1.10.3 Criterio de información de Akaike (AIC) y criterio de información Bayesiano (BIC).

Cuando se tiene una serie de modelos *M1*, *M2*,... con parámetros *K1, K2,…,* respectivamente, una metodología para compararlos corresponde a la función de máxima verosimilitud (*likelihood*). La máxima verosimilitud permite seleccionar el modelo que realiza el mejor ajuste de los datos pero no penaliza su complejidad, lo que si sucede cuando se emplean medidas de contraste como el *AIC* y el *BIC*. Ambos criterios hacen uso del *Log-likelihood*(*log Lik*), que es el logaritmo de máxima verosimilitud, y sustraen un término proporcional al número de parámetros ( ) en el modelo, así*: logLik -αK*, donde α corresponde a 2 para el *AIC* y a *log(N)* para el *BIC* (Posada and Noguera, 2007).

1.10.4 Criterio de información de Akaike (AIC)

Según (Posada and Noguera, 2007). El criterio combina la teoría de máxima verosimilitud, información teórica y la entropía de información, y es definido por la siguiente ecuación:

$$AIC = -2 * log\ Lik + 2K \quad (7)$$

Este criterio tiene en cuenta los cambios en la bondad de ajuste y las diferencias en el número de parámetros entre dos modelos. Los mejores modelos son aquellos que presentaron el menor valor de *AIC*.

Cuando los valores de *AIC* están muy cercanos, la escogencia del mejor modelo se puede realizar con base en el cálculo de la probabilidad (pesos de *Akaike*) y la probabilidad relativa (relación de evidencia), a través de las siguientes ecuaciones:

$$Probabilidad = \frac{e^{-0.5^{\Delta}}}{1 + e^{-0.5^{\Delta}}} \tag{8}$$

$$Probabilidad\ relativa = \frac{Probabilidad\ de\ que\ el\ modelo\ 1\ sea\ correcto}{Probabilidad\ de\ que\ el\ modelo\ 2\ sea\ correcto} = \frac{1}{e^{-0.5^{\Delta}}} \tag{9}$$

Donde, Δ es la diferencia entre los valores de *AIC*.

1.10.5 Criterio de información Bayesiano (BIC)

El *BIC* es calculado para los diferentes modelos como una función de la bondad de ajuste del *log Lik*, el número de parámetros ajustados ( ) y el número total de datos ( ). El modelo con el más bajo valor de *BIC* es considerado el mejor en explicar los datos con el mínimo número de parámetros. El *BIC* está definido por la ecuación: (Posada and Noguera, 2007).

$$BIC = -2 * log\ Lik + log(N) * K \tag{10}$$

Conclusiones parciales

- Es de gran importancia almacenar en bases de datos, los datos provenientes de la identificación para su posterior análisis o para ser usado en futuras investigaciones.

- Se empleará MySQL como sistema gestor de base de datos, pues es de fácil configuración, permite la comunicación con MATLAB y se encuentra muy referenciado su uso en aplicaciones ingenieriles, así como la herramienta gráfica MySQLWorkbench.

Capítulo 2. Funciones “SaveIntoDbNp” Y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

El desarrollo de la investigación parte de las consideraciones teóricas presentadas, en ellas quedan plasmados aspectos básicos dentro del ámbito de la identificación de procesos. Atendiendo a las necesidades que demanda actualmente la ingeniería en cuanto a la validación de modelos matemáticos, el proyecto responde a su objetivo general de crear las funciones para obtener información de modelos estimados y almacenarlos en la base de datos *MATLAB_DATA* de MySQL y el análisis de los datos obtenidos del proceso de identificación, por lo que se presenta la siguiente propuesta:

2.1. SaveIntoDbNp. Aspectos generales.

SaveIntoDbNp es una función que se pretende añadir a la extensa colección de funciones de MATLAB. Esta permite al usuario crear la base de datos relacional MATLAB_DATA en MySQL, si no existe, así como obtener la información que brinda MATLAB de modelos estimados seleccionados por el usuario y exportarla a dicha base de datos. Esta selección se realiza después de haber estimado dichos modelos con el uso del Toolbox de Identificación de Sistemas, mediante el la herramienta de Identificación de Sistemas GUI o a través de los comandos de MATLAB.

Con SaveIntoDbNp se crean tablas, y en cada uno de sus campos es almacenada información de interés sobre cada modelo seleccionado por el usuario.

Por ejemplo: Un usuario puede estimar la cantidad de modelos que desee con el Toolbox de Identificación de Sistemas y puede exportarlos a la base de datos MATLAB_DATA, después de conectado. Posteriormente, en el siguiente orden, se introducen como

argumentos de la función: la variable de conexión, las que contienen los identificadores del proceso y el lazo, la que contiene los datos de validación y las variables que se corresponden con los modelos estimados. Es importante señalar, que los argumentos deben introducirse en el orden especificado y todas las entradas deben estar separadas por comas. Si el procedimiento es correcto, se almacenan los datos en la base de datos en cuestión para un uso futuro.

Sintaxis de la función: saveintodbnp (conn,'proc','loop', val_data, model1, model2,…, modeln)

2.1.1. Características de la función SaveIntoDbNp

- Es una función primaria[2].
- En su diseño se utilizó el paradigma de programación estructurada.
- Escrita en lenguaje m de MATLAB y SQL.
- No es un paquete de funciones, es una función única que emplea funciones de la biblioteca de MATLAB.
- Es diseñada para exportar información a una base de datos de MySQL que puede ser creada por la función misma automáticamente (por defecto la base de datos se llama MATLAB_DATA). De otro modo puede ser creada por el usuario en el servidor MySQL.
- Permite la exportación de datos, al unísono, desde un puesto de trabajo local y/o puestos de trabajos remotos a través de la red.
- Las tablas necesarias son creadas por la función de forma predeterminada dentro de la base de datos MATLAB_DATA.
- Como se muestra en la primera línea de código ejecutable, en la cual la función es declarada: saveintodbnp (conn,'proc','loop', val_data, varargin[3]), la función

[2]*Función primaria: Es una función visible a otras funciones que están fuera del archivo en el cual ella está definida* (Mathworks, 2013).

admite como mínimo cinco entradas y no tiene ninguna salida. O sea, está concebida para al menos cinco entradas, donde no varía la variable de conexión, la variable que contiene el identificador del proceso al que pertenecen los modelos, la variable que contiene el identificador del lazo dentro del proceso y la variable que contiene el set de datos usados para la validación[4], sin embargo el número de modelos puede aumentar y con ello el número de entradas. Es necesario especificar, que en una inserción los modelos deben pertenecer a un mismo lazo y por ende a un mismo proceso.

Estas variables deben ser introducidas en un orden específico:

1) Variable de conexión.
2) Variable que contiene el identificador del proceso.
3) Variable que contiene el identificador del lazo.
4) Variable que contiene los datos de validación.
5) El modelo del cual se pretende almacenar información.
    - Solo almacena modelos lineales paramétricos y no paramétrico, de estructuras ARX, ARMAX, BJ, OE, Impulso y Respuesta al paso .
    - Por el momento la función está diseñada para sistemas SISO (*Single Input Single Output*).

[3]*La sentencia “*`varargin`*” es usada solo dentro de una función para contener argumentos de entrada opcionales pasados a la función. El argumento varagin tiene que ser declarado como la última entrada de la función, coleccionando todas las entradas desde ese punto en adelante* (Mathworks, 2013).
[4]*Set de datos usados para la validación: Son los datos usados para validar un modelo, comparándolo con la salida del modelo (model output) y aplicarle las pruebas de residuos (residual tests)* (Mathworks, 2013).

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

2.2. Inicialización

Para poner en marcha la función y sacar un buen provecho de ella es necesario haber instalado los *software* en los que va a ser usada y configurarlos.

2.2.1. Requerimientos para el uso de la función SaveIntoDbNp

1) Se requiere tener instalado el *software* MATLAB. La función ha sido probada en la versión 7.10.0.499 (R2010a) (win32) de este *software*.

2) Se precisa de un servidor MySQL. La función ha sido probada en la versión de servidor: 5.5.14 MySQL *Community Server* (GPL).

3) Con el objetivo de establecer una comunicación entre MATLAB y dicho servidor es indispensable instalar un controlador (*driver*) ODBC o JDBC. Driver probado: MySQL Connector/ODBC 5.1.

4) Tipos de datos.

La función *saveintodbnp* está diseñada para que maneje los siguientes tipos de datos:

- *Database*: Correspondiente a la variable de conexión. La clase *Database* provee una interfaz de comunicación con los archivos donde está almacenada la información con respecto a la conexión. Esta clase contiene propiedades y métodos para obtener información almacenada en la base de datos, relacionada con la definición de mensajes y señales (Mathworks, 2013).
- *Char*: Cadenas de caracteres correspondientes al identificador del lazo y del proceso.
- *Iddata*: Correspondiente con los datos de validación. Un objeto iddata representa datos de tiempo-dominio o frecuencia-dominio. Además, estos proveen una manera estándar de manejar datos en el *Toolbox* de Identificación de Sistemas (Mathworks, 2013).

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

Los siguientes corresponden con los modelos:

- *Idmodel* es una superclase para modelos lineales, es un objeto con el que no se trata directamente, pues contiene las propiedades comunes de los objetos modelos *idarx, idfrd , idproc, idpoly, idgreye idss*(Mathworks, 2013), siendo estos los de interés en la investigación, menos los dos últimos que no serán tratados.
- *Idpoly* modelos lineales polinomiales de entrada-salida. Crea un objeto modelo conteniendo parámetros que describen de forma general estructuras de múltiple entrada simple-salida (Mathworks, 2013). De interés estructuras ARX, ARMAX, BJ y OE.

2.2.2 Configurando el entorno

1) Ejecutar la función que se encuentra programada en un fichero **.m** de MATLAB.
2) Como cualquier función de MATLAB los datos que va a usar *saveintodbnp* deben estar en el *workspace*(espacio de trabajo). Si los modelos a exportar son generados a través de los comandos de MATLAB, estos se encuentran en el *workspace* por defecto. Si los modelos fueron generados en la herramienta de Identificación de Sistemas GUI, estos, y los datos de validación deben ser exportados al *workspace*, debido a que no están automáticamente disponibles en el *workspace*. Cada modelo se selecciona y por separado se pueden arrastrar al rectángulo ***To Workspace*** en la herramienta de Identificación de Sistemas GUI. Rápidamente el modelo aparece en el *workspace*. Ver Figura *2.1*.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

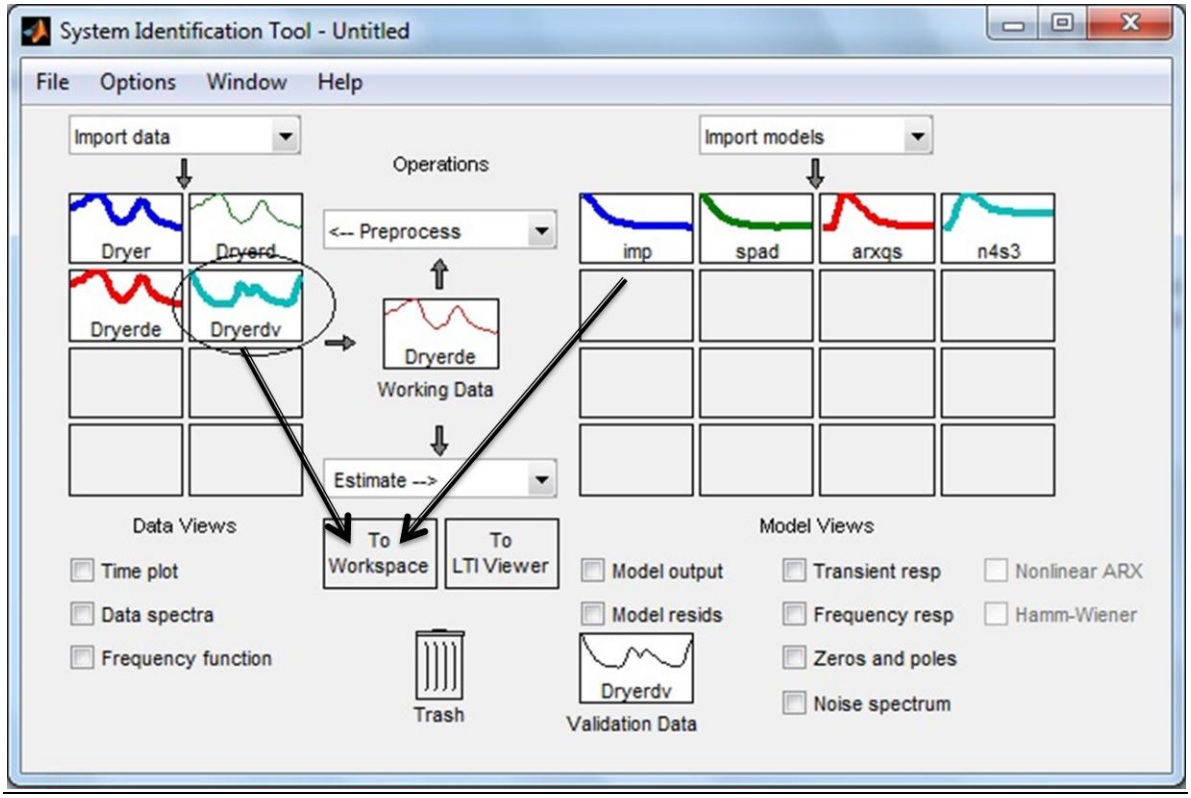


Figura 2.1Exportar a workspace desde el GUI del Toolbox de Identificación de Sistemas

3) Por último, después de haber importado los modelos y los datos de validación al workspace, se realiza la conexión con la base de datos a través del comando:

a) Para el uso con ODBC: conn = database ('datasourcename', 'username', 'password')

b) Para el uso con JDBC: conn = database ('databasename', 'username', 'password', 'driver', 'databaseurl')

En ambos casos el primer argumento es la fuente de datos, abordada posteriormente, el segundo y el tercer argumento son el usuario y/o contraseña requeridos para conectarse a la base de datos.

En el segundo caso el cuarto argumento es el tipo de controlador: JDBC, y el quinto es la estructura del JDBC URL.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

Luego de especificados cada argumento, en cada caso, la estructura de la conexión queda almacenada en la variable conn.

Se recomienda usar la función *logintimeout* antes de usar el comando *database*, para especificar el tiempo máximo en la que se trata de establecer la conexión con la base de datos.

Nota: Si se usa un controlador JDBC, no se necesita configurar una fuente de datos (*data source*). Solo se provee el controlador JDBC con la información necesaria sobre la localización de la base de datos, cuando MATLAB se conecta a la base de datos usando el comando database.
Para utilizar el *Toolbox* de Base de Datos con la base de datos y un controlador ODBC, se debe configurar una fuente de datos. Esta consiste en: *“Los datos a los que el Toolbox de Base de Datos se requiere que acceda, e información acerca de cómo encontrar estos datos, dígase un controlador, directorio, servidor, o nombres de redes”* (Mathworks, 1998).

En la configuración de la fuente de datos se asigna un nombre a cada base de datos. Pueden configurarse fuentes de datos locales, para bases de datos que residen en la PC donde se esté trabajando, o fuentes de datos remotas para bases de datos, que residen en sistemas conectados en red con la PC en que se está trabajando. Los procedimientos para configurar fuente de datos locales y remotos no difieren prácticamente. Para usar las instrucciones apropiadas en la configuración de la fuente de datos para el uso con controladores ODBC ver Anexo 7y con controladores JDBC ver Anexo 8.

2.3. Diseño de diagrama EER[5]

El diagrama EER es diseñado a partir de interrogantes como: ¿Qué datos se van a almacenar? ¿De qué forma están estructurados? y ¿Cómo se va a acceder a ellos? En este sentido son analizados a profundidad cada uno de los elementos descritos en el subepígrafe:

[5] Diagrama EER: Diagrama del Esquema Relacional del inglés *Enhanced Entity Relationships diagram.*

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

Fase 2. Definición y obtención de la información a almacenar de cada modelo Con el objetivo de definir las entidades y atributos, y establecer las relaciones entre dichas entidades.

Teniendo en cuenta tales demandas, se asumen las siguientes entidades y atributos para el diseño del diagrama EER, ver *Tabla 2.1*.

Tabla 2.1 Entidades-Atributos

#	Entidad	Atributos											
1	Modelo	proceso	lazo	fecha	estructura	nombre	FIT	na	nb	nc	nd	nf	nk
2	Gráfica respuesta al paso		valores de la abscisa				valores de la ordenada						
3	Gráfica respuesta de frecuencia	valores de la abscisa	valores de la ordenada del 1er gráfico, relacionados con la amplitud				valores de la ordenada del 2do gráfico, relacionados con la fase						
4	Gráfica análisis de residuos corr.		valores de la abscisa				valores de la ordenada						
5	Gráfica análisis de residuos corr.-cruzado		valores de la abscisa				valores de la ordenada						
6	Vector		tipo de vector				valor						
7	Ceros y polos		parte real				parte imaginaria						
8	Gráfica de espectro de ruido		valores de la abscisa				valores de la ordenada						

Para cada una las entidades se crean una tabla con motor de almacenamiento (*engine*) InnoDB[6]. Sus atributos se convierten en campos de dichas tablas, ver *Figura 2.2 Diagrama EER de la base de datos MATLAB_DATA*. Por otra parte, con el propósito de especificar el tipo de datos de cada campo, se analiza el tipo de estructura de cada atributo, por ejemplo: El atributo nombre es una cadena de caracteres, por lo que se especifica en el campo *name* el tipo de datos (*varchar*), ver *Figura 2.2.*

[6]*InnoDB: Es el motor de almacenamiento más popular, diseñado para el procesamiento de transacciones, específicamente procesamiento de transacciones cortas, la cuales usualmente se completan* (Schwartz et al., 2012).

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

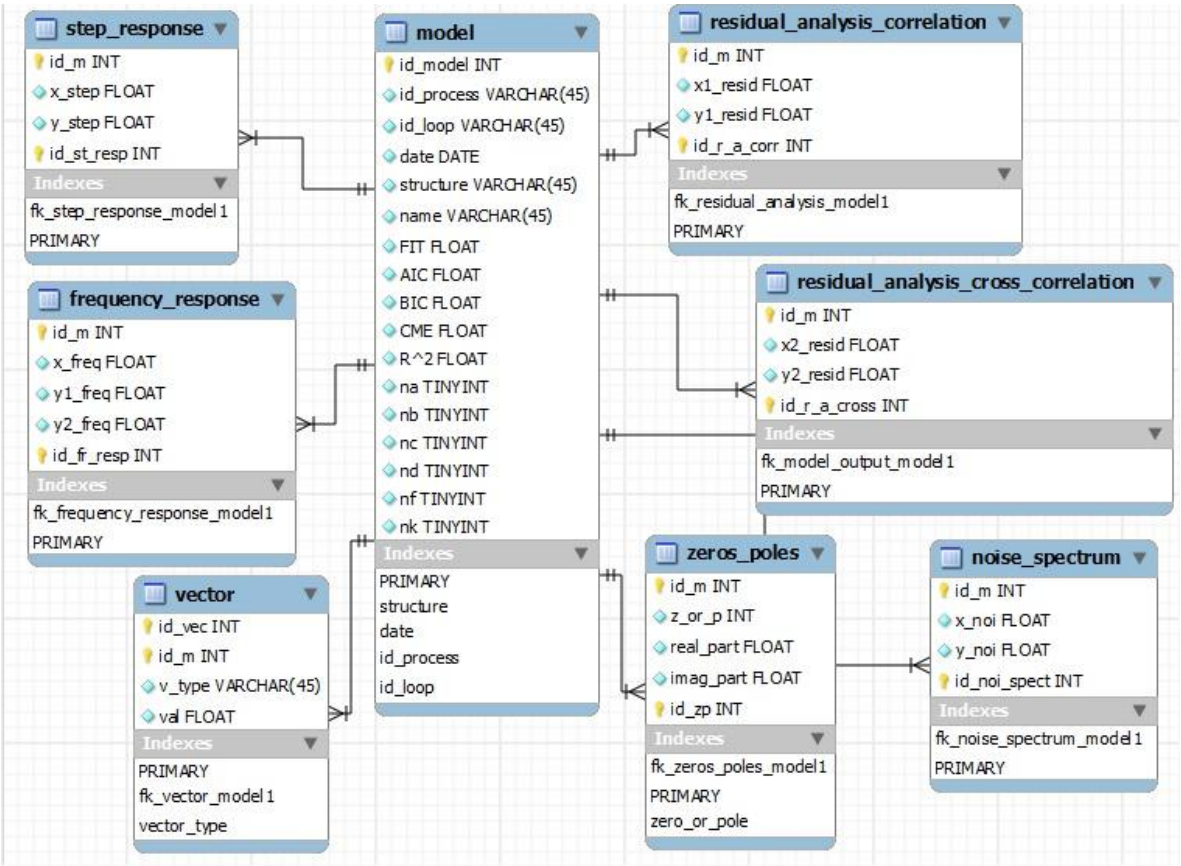


Figura 2.2 Diagrama ERR de la base de datos utilizada.

2.4 Diagrama de planificación. Explicación de la programación de la función *SaveIntoDbNp* por fases.

El diagrama de actividades fue concebido en cuatro fases, en cada de una de ellas fueron agrupadas las tareas de la función de acuerdo a su orden lógico.

Nota: Durante la programación de la función se utilizó como planta a modelar el Secador de pelo (*Dryer*) encontrado dentro de los *demos* del propio MATLAB y su juego de datos de entrada-salida.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

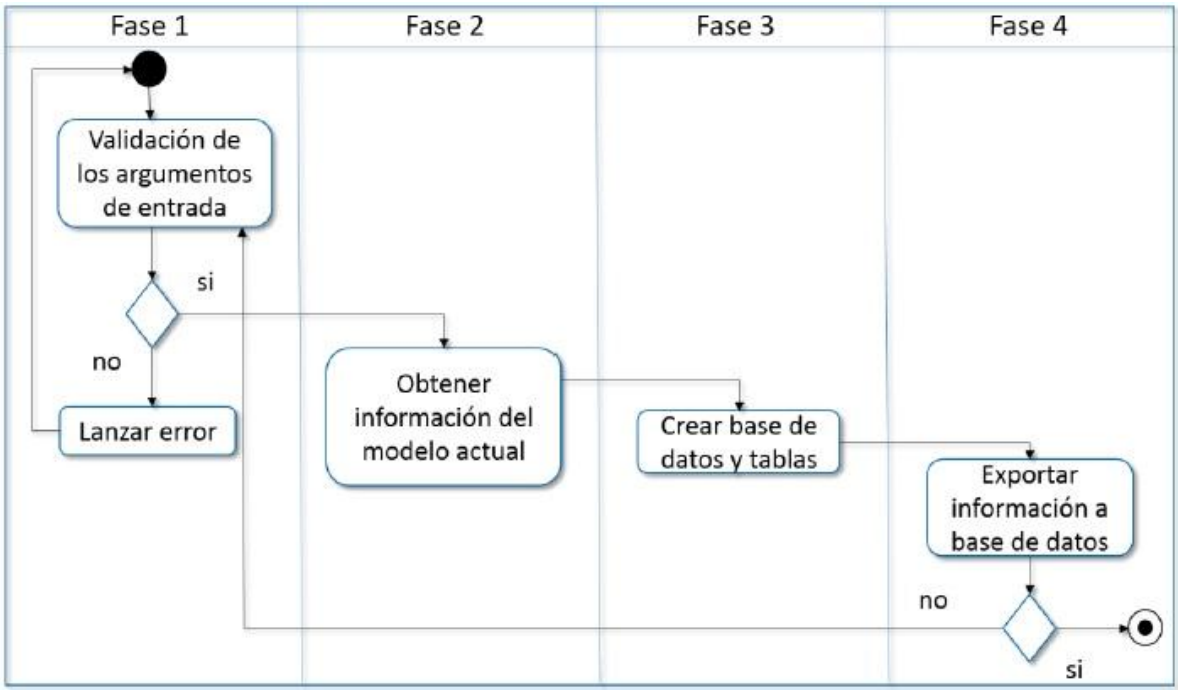


Figura 2.3 Diagrama de planificación de la función SaveIntoDbNp

2.4.1 Fase 1: Validación de los argumentos de entrada pasados a la función *SaveIntoDbNp.*

En la primera línea de código ejecutable se declara la función S*aveIntoDbNp*:

```
saveintodbnp(conn,'proc','loop',val_data, varargin)
```

El primer argumento de entrada (*conn*) debe coincidir con la variable de conexión mencionada anteriormente. El segundo y tercer argumento debe corresponder con el proceso y el lazo al que pertenecen los modelos a exportar en una inserción. El cuarto argumento de entrada (*val_data*) debe corresponderse con la variable que contiene los datos de validación. El quinto argumento, *varagin,* permite la entrada de múltiples modelos.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

Para el correcto funcionamiento de la función saveintodbnp se garantiza que los argumentos de entrada sean lo más fiable posible por lo que se lleva a cabo el siguiente proceso de validación de los datos.

1- Validación del primer, segundo, tercer y cuarto argumento.

Primero se comprueba que hayan más de cuatro entradas, de lo contrario, el programa emite un error explicando que no son suficientes entradas. Si hay más de cuatro entradas, se calculan la cantidad de modelos introducidos, y luego se verifica cada entrada para validarlas según sea su clasificación.

Se comprueba que la primera entrada (variable de conexión) sea del tipo *"database"* y que la conexión exista. Posteriormente se chequea la segunda y tercera entrada: deben ser del tipo *"char"*, pues contienen el proceso y el lazo al que pertenecen el grupo de modelos a exportar respectivamente. Luego se verifica la cuarta entrada (variable que contiene el set datos usados para la validación del modelo) que debe ser del tipo *"iddata"*. Si alguna de las entradas no son del tipo especificado, el programa emite un error. Si las entradas son del tipo requerido, el programa sigue su curso.

2- Validación de los demás argumentos. (Dependerá de la cantidad de modelos introducidos).

Se realiza un ciclo *for* a partir de la cantidad de modelos introducidos que fue calculada. Esto ocurre con el objetivo de verificar el tipo de datos de cada supuesto modelo. Si el supuesto modelo no presenta una de las estructuras especificada por las características de la función, este no es exportado a la base de datos. Cuando cuenta con la estructura especificada, el modelo sí es exportado. O sea, dentro del ciclo *for* se valida o no un modelo, de no ocurrir la validación, se emite un alerta (*warning*) reportando el modelo que no fue validado. De ocurrir la validación, este modelo se convierte en "modelo actual". Para

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

el trabajo con este modelo actual, se crea una variable temporal a la cual, él es asignado, denominada *current_model*.

2.4.2 Fase 2. Definición y obtención de la información a almacenar de cada modelo.

Cuando se emplea el término guardar, o almacenar un modelo, se refiere a exportar a la base de datos cierta información de ese modelo que es imprescindible para análisis posteriores. O sea, con esta información se crean experiencias que pueden ser usadas por otros programas en un futuro para comparar dichos modelos, análisis estadísticos, toma de decisiones, etc.

La información que se exporta a la base de datos está conformada por los siguientes elementos:

- El identificador del proceso al que pertenecen los modelos. Es introducido por el usuario como argumento de la función.
- El identificador del lazo dentro del proceso: Es pasado por el usuario como argumento de la función.
- La fecha en que es guardado el modelo: El servidor facilita una fecha y un formato de fecha global para sus usuarios, lo cual permite eliminar confusiones, pues las PCs de los diferentes usuarios pudieran tener establecidos diferentes fechas y/o formato de fechas. De esta forma MATLAB obtiene la fecha que proporciona el servidor mediante la sintaxis fetch(conn, 'SELECT curdate()'). Donde el primer argumento es la variable de conexión y el segundo es una consulta de SQL que devuelve la fecha actual.
- Nombre: Se obtiene a través del uso del caracter “**.**” que asociado a una variable permite acceder a sus propiedades, por ejemplo: *current_model**.**name*
- FIT: Se obtiene mediante la función: [YH,FIT,X0]=compare(*val_data*,*current_model*), FIT constituye una salida de la función compare.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

- La información sobre la respuesta al paso (*step response*), respuesta de frecuencia (*frequency response*), análisis de residuos (*residual analysis*) y análisis del espectro de ruido (*noisespectrum*) de un modelo, es de gran interés en la pesquisa. Dicha información se obtiene mediante las gráficas generadas por las funciones step, bode, resid. Para recopilar estos datos basta con acceder a los registros donde MATLAB los almacena. Esto es viable a través de los *handles*(manejadores de cada figura), con el objetivo final de obtener los pares de puntos, que conforman cada una de las curvas representadas en las gráficas.

De manera general se explicará las líneas de código que en su conjunto tienen la función de extraer del MATLAB la información mencionada anteriormente:

La figura ***Step Response*** generada mediante la función *step*, está compuesta sólo por una única gráfica: abscisa (*x_step*) y ordenada (*y_step*).

```
43-step(current_model);
44- h=gcf;
45- h1 = get(h,'children');
46- h11 = get(h1,'children');
47- x_step = get(h11(1),'XData')';
48- y_step= get(h11(1),'YData')';
49- close(gcf);
```

Como se observa en la línea 43 se grafica *step response*, con el objetivo de obtener los pares de puntos mencionados previamente.

En la línea 44 se obtiene el *handle* de la figura actual (*step response*), para acceder a los registros donde MATLAB almacena los datos graficados en dicha figura. Con el empleo de este *handle*:

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

Se obtienen los hijos (*children*) de la figura (línea 45): en este caso es una única gráfica la que constituye a *step response.*

Se obtienen los hijos de esta gráfica (línea 46) que son los que permiten acceder a los vectores que conforman cada uno de sus ejes. Estos vectores: XData y YData, se crean en MATLAB por defecto y contienen los valores de la abscisa y la ordenada respectivamente. Asimismo, la información que contienen XData y YData se le asigna a las variables *x_step* y *y_step* respectivamente, para luego exportarla a la base de datos (líneas 47 y 48). Por último en la línea 49, se cierra la figura actual (*step response*).

La figura ***NoiseSpectrum*** generada mediante la función *bode*, está compuesta sólo por una única gráfica: abscisa (*x_noise*) y ordenada (*y_noise*), el algoritmo para la obtención de los valores de sus ejes es prácticamente igual al de la figura de *Step Response*.

La figura ***Frequency Response*** generada por la función *bode*, se conforma por dos gráficas: en la primera es computada la magnitud y en la segunda la fase de la respuesta de frecuencia de un modelo. Ambas se componen por la abscisa (*x_frequency*) y las ordenadas (*y1_frequency*) y (*y2_frequency*) de la primera y segunda gráfica respectivamente.

```
53-bode(current_model)
54-h=gcf;
55-h1 = get(h,'children');
56-h11 = get(h1(1),'children');
57-x_frequency = get(h11(1),'XData')';
58-y1_frequency= get(h11(1),'YData')';
59-h12 = get(h1(2),'children');
60-y2_frequency = get(h12,'YData')';
61-close(gcf);
```

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

El algoritmo funciona de forma similar al explicado anteriormente con la figura *step response*, con la diferencia de que ahora la figura actual es *frequency response* y que se obtienen dos gráficas en vez de una.

Cuando se obtienen los hijos de esta figura (línea 55), se obtienen las dos gráficas mencionadas con anterioridad.

Primero se obtienen los hijos de la primera gráfica (línea 56) para tener acceso a sus vectores XData y YData y así asignar la información a *x_frequencyyy1_frequency* respectivamente (líneas 57 y 58).

Luego se obtienen los hijos de la segunda gráfica (línea 59) para tener acceso al vector YData y asignar la información a *y2_frequency* (línea 60). No es necesario acceder al vector XData, pues coincide con el obtenido en la primera gráfica.

La figura ***Residual Analysis*** generada por la función *resid*, se compone por dos gráficas: *correlational residual analysis*(análisis de residuos correlacional) y *cross correlational residual analysis*(análisis de residuos correlacional-cruzado). Cada una de ellas está constituida por una abscisa y una ordenada: (*x1_resid, y1_resid)* y (*x2_resid, y2_resid)*.

```
65-resid(current_model,val_data)
66-h=gcf;
67-h1 = get(h,'children');
68-h11 = get(h1(1),'children');
69-x2_resid = get(h11(2),'XData')';%%residual_analysis_cross_correlation
70-y2_resid= get(h11(2),'YData')';
%%-------------------------------------
72-h12 = get(h1(2),'children');
73-y1_resid = get(h12(2),'YData')';%%residual_analysis_correlation
74-x1_resid = get(h12(2),'XData')';
75-close(gcf);
```

Se sigue la misma metodología del caso anterior, con la diferencia de que es ahora *Residual Analysis*la figura actual (línea 66). Además en las gráficas que componen dicha figura no

coinciden las abscisas, por lo que se accede a XData de ambas gráficas. Por último, a cada una de las variables *x1_resid, x2_resid, y1_resid* y *y2*_resid se le hace corresponder con su XData y YData (líneas 74, 69, 73 y 70).

2.4.3 Fase 3. Creación de base de datos y tablas desde MATLAB

Luego de diseñado el diagrama EER en Workbench, se genera el código en el lenguaje SQL, que recrea cada uno de los elementos y características de dicho diagrama. Como se pretende que la creación de la base de datos sea transparente al usuario, se incluye dentro del código de la función *SaveIntoDbNp* las sintaxis de SQL generadas por el Workbench. Esto es viable a través del uso de la función *exec* de MATLAB. Dicha función permite ejecutar sintaxis de SQL en un servidor MySQL desde MATLAB.

Aunque pareciera que cada vez que se ejecuta la función *SaveIntoDbNp* se crea la base de datos MATLAB_DATA y cada una de sus tablas, es necesario aclarar que esto no sucede. Dentro del código SQL se especifica que las tablas se crean si no existen con anterioridad, por ejemplo: 'create table if not exists MATLAB_DATA.model'.

2.4.4 Fase 4. Procedimiento para el almacenamiento de datos a las tablas

Es importante garantizar que el proceso de inserción en las tablas ocurra con fiabilidad; pues es imperativo dar la garantía de que se hayan almacenado todos los datos de un modelo, y no, que se almacenen a medias. Por ejemplo: No es práctico almacenar la mitad de los pares de puntos de una de las curvas, pues para un correcto análisis posterior, se necesitan todos los pares de puntos de dicha curva de lo contrario no sería efectivo dicho análisis. Esta condición conduce, a que se necesita concluir un grupo de sentencias que logren consumarse todas, o fallar como una unidad.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

Para lograr lo planteado anteriormente se usa un algoritmo llamado transacción. El cual se puede definir como*: “Un grupo de consultas (queries) que son tratadas atómicamente*[7]*, como una sola unidad de trabajo. Si el DBMS puede aplicar el grupo entero de consultas a la base de datos, lo hace, pero si no puede aplicar alguna de ellas, por un choque (crash) u otra razón, ninguna de ellas es aplicada. Es todo o nada”(SCHWARTZ, 2012).*

En consecuencia de lo explicado anteriormente, para el diseño de la base de datos se concibieron motores de almacenamiento InnoDB, como fue mencionado en la sección 2.3. Este motor de almacenamiento constituye la opción probada más estable y mejor integrada, hasta el momento(SCHWARTZ, 2012), lo cual nos brinda transacciones seguras.

Según DuBois (2002) una transacción se puede lograr manipulando el modo auto-comprometer (*auto-commit*) de MySQL, para permitir transacciones de múltiples sintaxis y luego consumarlas o no mediante *commit* o *rollback* en cada caso; estos términos y su filosofía de trabajo se explican a continuación.

En MATLAB esto es posible mediante el uso de las sentencias *try-catch* y las APIs *commit* y *rollback,* si la ejecución de *try* emite algún error se ejecuta *catch*.

2.4.4-1 Tabla *model*

El procedimiento para exportar a la tabla *model* se basa en la función *insert*(línea 175), el primer argumento de esta función es la variable de conexión, el segundo especifica la tabla en la que se va a almacenar, el tercero: la variable que contiene la relación de campos y el cuarto los datos a exportar. Previamente se almacena en la variable *exdata*(línea 171), la información referida a los datos que se desean exportar y en la variable *colnames*(línea 172), cada campo que almacena esta información.

[7]*Atomicidad: Una transacción tiene que funcionar como una simple e indivisible unidad de trabajo. La transacción completa es aplicada (commit) o no (rolled back). Cuando una transacción es atómica no existe algo como: Una transacción parcialmente completada (es todo o nada).* (Schwartz et al., 2012).

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

```
171- exdata = {proc, loop ,current_date, structure ,current_model.name, FIT, current_model.K, current_model.Tp1, current_model.Tp2 , current_model.Tp3, current_model.Tz , current_model.Td};
172- colnames = {'id_process','id_loop','date', 'structure' ,'name', 'FIT', 'K', 'Tp1', 'Tp2', 'Tp3', 'Tz','Td'};
175- insert(conn, 'model', colnames , exdata);
```

Partiendo de la estructura de la tabla *model*, es necesario retomar observaciones ya referidas, en cuanto a que la columna *id_model* es llave primaria de esta tabla. Por tanto, después de insertado un modelo en la tabla *model*, es decir, luego de insertado un registro se obtiene el id de este modelo. Esto último se logra mediante la función *fetch(conn, 'SELECT MAX(id_model) frommodel')*, el primer argumento es la variable de conexión y el segundo es una consulta de SQL que devuelve el mayor valor del campo *id_model*, o sea accede al id del modelo que acaba de ser insertado. De esta forma queda almacenado en la variable *last_id* el valor del id del modelo actual, así este valor se inserta en las demás tablas, especificando qué información pertenece a un mismo modelo, lo cual demuestra las relaciones antes explicadas.

2.4.4-2 Tabla *step_response*

Para esta tabla se usa un algoritmo similar al de la tabla *model*, sin embargo, difieren en que al exportar a la tabla *step_response*, se insertan múltiples registros en una inserción. En este sentido el número de registros se corresponde con el número de pares de puntos de la gráfica respuesta al paso.

Con la finalidad de exportar los datos a la tabla *step_response*, se hace coincidir el vector *x_step* y *y_step* con el campo del mismo nombre en la tabla. Para especificar que todos estos datos pertenecen a un mismo modelo se crea la variable *current_id_model*(línea 182). Esta es igualada al vector *x_step*, con el objetivo de que current_id_model, ahora vector, cuente exactamente con la misma longitud de *x_step* y *y_step*. Cada uno de los valores de

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

current_id_model, es remplazado por el contenido de la variable *last_id*(línea 185), la cual contiene el id del modelo al que pertenecen estos datos.

```
182- current_id_model=x_step;
183- elements = length(current_id_model);
184-for  n=1:elements
185- current_id_model(n)=last_id;
186-end
187- pre_exdata = {current_id_model , x_step , y_step};
188- exdata = cell2mat(pre_exdata);
```

Al emplear la función *insert*, se hace coincidir el vector *current_id_model* con el campo *id_m*, pues la tabla *model* está relacionada con la tabla *step_response* como se explica en el diagrama EER. El campo mencionado anteriormente está relacionado con el campo *id_model* de la tabla *model*, pues este segundo es llave foránea del primero, lo que justifica que funcione como índice en la tabla *step_response*.

Al igual que cuando se insertan datos en la tabla *model*, se usa *exdata* como variable de datos, pero como es explicado anteriormente, en este caso se introducen de una vez múltiples registros: tres vectores en forma de columnas (línea 187). Estos vectores antes de ser insertados, son convertidos a una matriz mediante la función *cell2mat* (línea 188), así lo requiere la función *insert* para insertar múltiples registros a la vez.

2.4.4-3 Tabla *noise_spectrum*

Se usa el mismo procedimiento del caso anterior, solo que *current_id_model* es igualado a *x_noise*, además se hace coincidir las variables *x_noise* y *y_noise* con los campos que llevan el mismo nombre.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

2.4.4-4 Tabla *frequency_ response*

Se usa la misma metodología del caso anterior, solo que *current_id_model* es igualado a *x_frequency*, además se hace coincidir las variables *x_frequency*, *y1_frequency* y *y2_frequency* con los campos que llevan el mismo nombre.

2.4.4-5 Tablas *correlational_residual_analysis y cross_correlational_residual_analysis.*

Para insertar los datos en estos gráficos, se sigue la metodología usada en los casos previos. Se reevalúa *current_id_model* para cada caso y se hacen coincidir las variables *x1_resid*, *y1_resid* (para la primera tabla) *x2_resid, y2_resid* (para la segunda tabla) con las columnas con el mismo nombre.

2.4.4-6 Tabla *zeros_poles*

Se utiliza el procedimiento expuesto en los casos anteriores referido a la forma de insertar la información. Solo que antes de obtener la matriz final que posee los datos a exportar se crean dos matrices previas, la primera contiene los ceros y la segunda contiene los polos. Para especificar si el registro contiene datos de un cero (se inserta un 0) o si contiene datos de un polo (se inserta un 1) es usado el índice *z_or_p*.

2.4.4-7 Tabla *vector*

En esta tabla la metodología cambia en cuanto a la forma de insertar la información, pues los campos no tienen el mismo tipo de datos. Específicamente el campo *v_type* es usado para almacenar cadenas de caracteres (siendo estas el nombre de los vectores que devuelve la función *polydata*), mientras que los demás campos son enteros o decimales. Esto significa que no se puede crear una matriz con los datos a almacenar, como se hace en los casos anteriores.

En lugar de insertar una matriz con los datos, se inserta un registro a la vez, por ejemplo: el primer valor del vector A, el nombre del tipo de vector al que pertenece ('A') y el id del modelo al que pertenece. Luego el segundo valor de este mismo vector con los restantes

campos, y así sucesivamente, hasta que se almacenen todos los valores de este vector para comenzar con el siguiente, en tanto se almacene el último valor del último vector.

Todo este mecanismo de inserción en las tablas ocurre dentro de la última fase del diagrama de actividades. A partir del cual, al final de esta cuarta fase ocurre una bifurcación, pues si no han sido almacenados todos los modelos, se retorna a la primera fase. Esto es resultado del ciclo *for* mencionado en la fase uno, el cual se realiza a partir de la cantidad de modelos introducidos que fue calculada. Es decir, hasta que no se almacene el último modelo introducido como argumento en la función *saveintodbnp*, dicha función no llega a su estado final, en el cual se concluye la comunicación entre MATLAB y la base de datos *MATLAB_DATA*.

2.5 Generalidades de la función QIdem

El propósito general que trae consigo dicha función, es la de facilitar dentro del proceso de identificación, la selección del mejor modelo para una aplicación determinada y por ende esta aplicación es destinada a usuarios que dominen los principales elementos relacionados con la identificación.

Para obtener respuestas más acertadas, está el referente a la adecuada selección de la estructura de los modelos, con el fin de la identificación de un sistema real, existen diversos tipos de estructuras de modelos, siendo necesario que el usuario determine la estructura que necesita para determinar su modelo.

La función permite la selección de todos los criterios o solo algunos, pero es importante tener en cuenta, que cada criterio mide diferentes características del modelo, por lo que la selección del orden del modelo que más se adecua a la realidad solo se puede realizar dentro de los resultados obtenidos para un mismo criterio.

Para la determinación del modelo y el cálculo de los criterios de bondad de ajuste se siguen los siguientes pasos:

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

- Dividir los datos reales en dos partes; una destinada a la identificación y la otra a la verificación del modelo obtenido.
- De acuerdo al tipo de modelo se calcula este para el *orden* requerido.
- Se determinan los criterios *AIC, BIC, CME* y *R2* correspondiente al modelo y al *orden* identificados.

Al concluir el proceso se dispondrá de la información acerca del valor de los índices para cada estructura de modelo y *orden* especificado, pudiéndose seleccionar dentro de una estructura y criterio de bondad cual es el *orden* que mejor se ajusta, que es el que menor valor del criterio ofrece.

2.5.1 Validacion de la función QIdem

Seguidamente la función, procederá a implementar el cálculo de los criterios de bondad de ajuste, para los cuales se necesitan datos provenientes de los modelos como es el caso de la suma de los ordenes estimados del modelo, el tamaño de la muestra, la función logLik, el error obtenido mediante la función pe, los cuales fueron obtenidos por los algoritmos que se muestran a continuación en un ejemplo para el modelo arx1:

```
AICarx=aic(marx);        %Cálculo del criterio
AICdarx=ma(i,1)+ma(i,2)+ma(i,3);%Suma de los parámetros estimados del modelo
N=length(y);                  %Tamaño de la muestra
LFarx=darx-AICarx/2;          %Obtención de la función logLik
[AICarx,BICarx]=aicbic(LLFarx,darx,N)  %Cálculo del BIC
earx=pe(marx,planta);         %Cálculo del error mediante la función pe
```

Esta secuencia se muestra como se obtienen los valores correspondientes a una parte de los criterios, para ello se encontró entre las funciones de MatLab, la función “aic”, esta permite obtener el valor del criterio AIC luego de estimarse el modelo. Además este coeficiente fue utilizado para a partir de las ecuaciones planteadas en el capitulo anterior, obtener el valor correspondiente a la función logLik, quedando como se muestra en la secuencia de

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

códigos, está función fue necesaria obtenerla para luego proceder a calcular el valor del BIC, mediante una función perteneciente al MatLab.

El error que se muestra es calculado por una función llamada *pe* la cual da como resultado la varianza de los residuales según la muestra, el cual se utilizará para el cálculo de la suma de los cuadrados de las distancias verticales de los puntos desde la curva de los residuales.

Esta secuencia está sucedida por un ciclo *for*, encargado de calcular la suma de los cuadrados del error (SCE), la que será utilizada para obtener el valor del CME y del R^2. Mostrados seguidamente como continuación del ejemplo del modelo arx1:

```
SCEarx=0;                                %Inicializa SCE en cero
for v=1:1:length(y)                    %Ciclo for paradeterminar SCE
SCEarx=SCEarx+earx(v)*earx(v);
end
CMEarx=SCEarx/(length(y)-darx) %Algoritmo para el cálculo de CME
ysarx=y'-earx;                  %Valores de salidas estimados
ymarx=mean(ysarx);              %Valores medios de y
rvmyarx=y-ymarx;
SCTarx=0;                       %SCT inicializado en cero
forns=1:1:length(y)
SCTarx=SCTarx+rvmyarx(ns)*rvmyarx(ns); %Calculo de SCT

end
        R2arx=1-(SCEarx/SCTarx) %Cálculo del criterio del determinante R2
```

Siendo:

- SCEarx1: Suma de los cuadrados del error, la cual será utilizada para calcular el Cuadrado Medio del Error y el criterio del Determinante.
- CMEarx1: Cuadrado medio del error calculado mediante la SCE dividido la resta del tamaño de la muestra y el número de órdenes estimados del modelo.
- ysarx1: salida estimada del modelo.
- ymarx1: valor medio de la salida real.
- rvmyarx1: resta de la salida real y del valor medio de la misma.

Funciones “SaveIntoDbNp” y “QIdem”, para el almacenamiento y análisis de los datos obtenidos en la identificación “No paramétrica” con Ident del MatLab.

- SCTarx1: Suma de los cuadrados totales utilizada para el cálculo del criterio del Determinante.
- R^2: Criterio del determinante, para el cual se utiliza la SCE y la SCT, y su resultado esta expresado en porciento.

2.6 Conclusiones parciales

- La estructura de la base de datos (reflejada en el diagrama EER) es simple y no admite valores nulos dentro de las tablas, lo cual garantiza en gran medida la integridad de los datos
- La función prevé los errores más comunes que pudiera cometer el usuario y advierte mediante un grupo de alertas, que le indican la naturaleza de dichos errores en cada caso.
- Los criterios de validación programados son los más utilizados en aplicaciones técnicas.
- La función permite la obtención de diferentes criterios de validación, pero es importante tener en cuenta, que cada criterio mide diferentes características del modelo.

Capítulo 3. Prueba y análisis de los resultados

3.1 Naturaleza de los datos

Con el objetivo de realizar pruebas a la función "saveintodbnp" y comprobar su correcto funcionamiento, se pretende utilizar un juego de datos que nos brinda el MatLab. Para ello se escogió el juego de datos provenientes del secador (Dryer).

3.2Procedimiento para el uso de la función

Luego de identificado los modelos, estos se deben exportar al workspace de MATLAB. Posteriormente, mediante el uso de la función saveintodbnp pueden ser guardados en la base de datos. Lo descrito anteriormente se logra a través de los siguientes pasos:

Paso 1. Cargando los datos a exportar en el workspace de MATLAB.

Si el proceso de identificación fue llevado a cabo a través de los comandos de MATLAB, los modelos y los datos de validación se encuentran en el workspace por defecto.

Si los modelos fueron generados en la herramienta de Identificación de Sistemas GUI, los mismos y los datos de validación deben ser importados al workspace.

Nota: Para una mayor información, consulte el subepígrafe 2.2.2

Paso 2. Creando la conexión con la base de datos

Mediante el comando database se crea la conexión con la base de datos.

Nota: Para una mayor información, consulte el subepígrafe 2.2.2

Paso 3. Llamando la función

Se invoca la función mediante su sintaxis. En este paso se le introducen todos los argumentos requeridos y los modelos que se deseen guardar.

Nota: Para una mayor información, consulte el subepígrafe 2.1.1

3.2.1 Ejemplo del uso de la función saveintodbnp y comprobacion de los datos

Se importa a la herramienta de Identificación de Sistemas GUI el juego de datos entrada-salida provenientes del ejemplo que nos brinda el MatLab, perteneciente al secador (Dryer). Luego de tratado este juego de datos, se estiman cuatro modelos, como se muestra en la *Figura 3.1*. Posteriormente se exportan al workspace (ver *Figura 3.2*) y se crea la conexión con el fin de utilizar la función saveintodbnp.

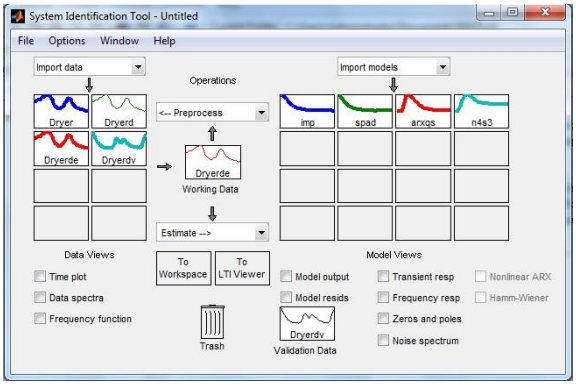
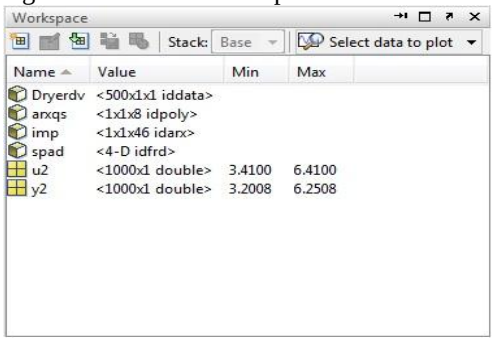


Figura 3.1 Modelos estimados a partir del juegos de datos Dryer.

Figura 3.2 Workspace del MatLab



.

A continuación se muestra el uso de la función, a la cual se le introducen la variable de conexión (conn), los identificadores del proceso y el lazo (proc y loop respectivamente), también los datos de validación (Dryerdv) y los cuatro modelos identificados (imp, spd, arxqs, n4s4).

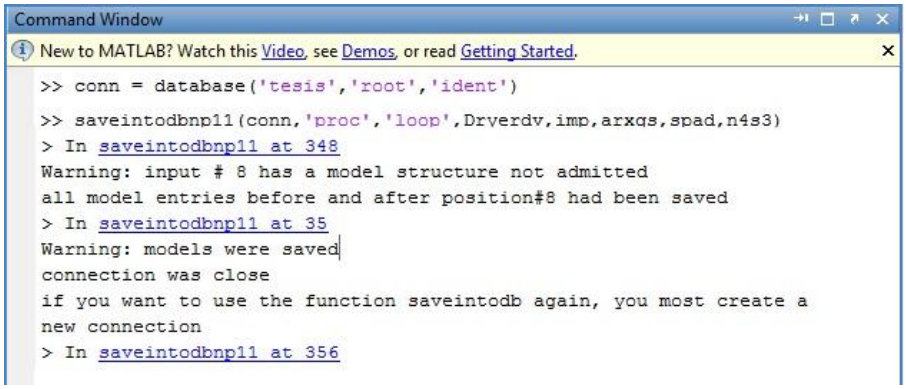


Figura 3.3 Uso de la función saveintodbnp en la ventana command del MatLab.

Como se puede apreciar (en la *Figura 3.3*) el intento de insertar el cuarto modelo (n4s4) falla y la función emite una alerta, pues como muestra la Figura 3.4 este es un modelo lineal de espacio estado, para el cual la función no está diseñada.

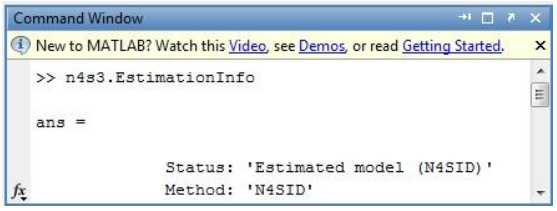


Figura 3.4 Modelo n4s3: modelo lineal de espacio-estado.

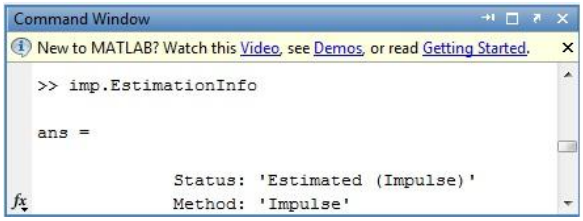


Figura 3.5 Modelo imp de estructura Impulse

También se observa en la *Figura 3.3* que los demás modelos sí fueron exportados "supuestamente", y que la conexión fue cerrada. El resto de los modelos sí presentan estructuras admitidas por la función.

Con el propósito de verificar que los pares de puntos que conforman las curvas de las gráficas: Respuesta al paso, respuesta de frecuencia, espectro de ruido y análisis de residuos, coinciden con sus tablas correspondientes, se grafica cada uno de estas gráficas en MATLAB, usando los comandos del *Toolbox* de Identificación de Sistemas como muestra la *Figura 3.6.* Se toman algunos puntos de ejemplo en cada curva y se muestran los registros correspondientes a cada uno de esos puntos.

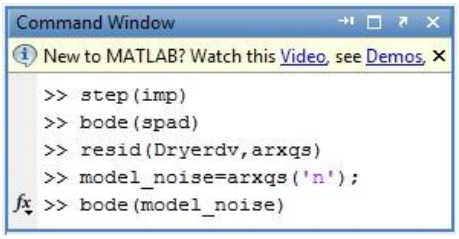


Figura 3.6 Funciones del Toolbox de identificación de Sistemas que generan gráficas de interés.

Como se puede observar en el campo id_m de la Tabla 3.1 está almacenado el valor "diez" lo cual indica que estos valores pertenecen al modelo arxqs. También se puede apreciar, en esa misma tabla, que los puntos coinciden con los mostrados a través de la identificación no paramétrica.

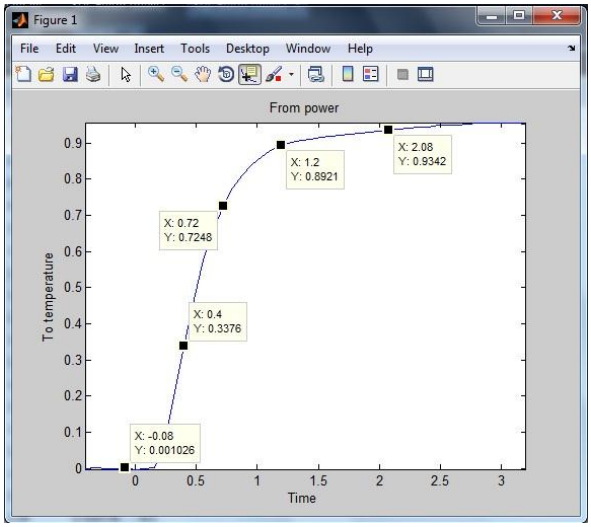


Figura3.7 Gráfica de la respuesta al paso del modelo no paramétrico (imp).

id_m	x_step	y_step	id_st_resp
10	-0.48	0	399
10	-0.4	0	400
10	-0.08	0	404
10	0.4	0.332632	410
10	0.72	0.720661	414
10	1.2	0.888821	420
10	2.08	0.920776	431

Tabla3.1 Segmento de la tabla step_response de la base de datos procedente del modelo estimado arxqs.

Los análisis que pueden realizarse con respecto al resto de las gráficas y sus tablas correspondientes, demuestran que para estas ocurre lo mismo que en el caso presentado anteriormente. A partir de estas conclusiones sólo se muestran a continuación.

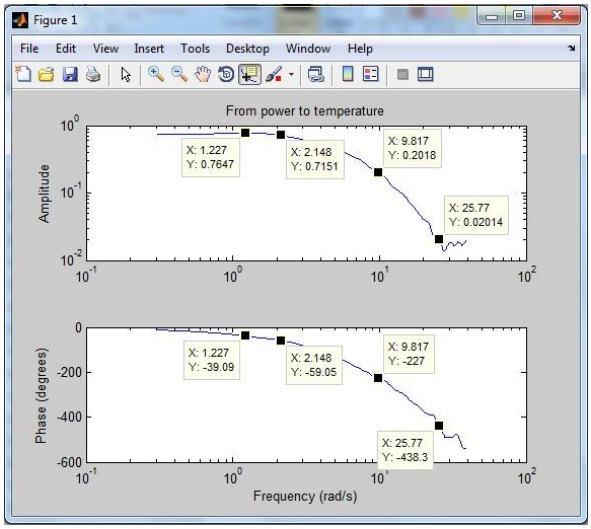


Figura 3.8 Gráfica de respuesta de frecuencia del modelo no paramétrico spad.

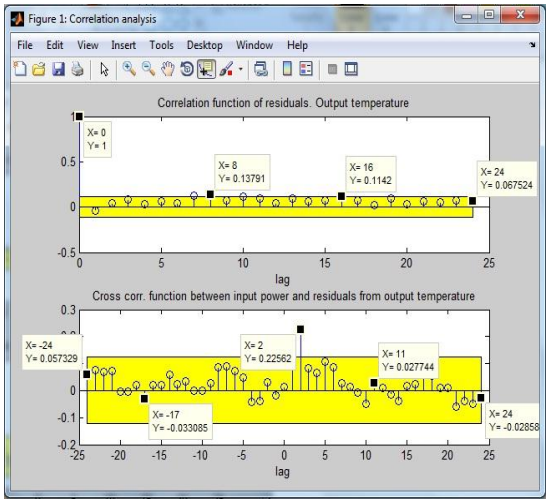


Figura 3.9 Gráfica de análisis de residuos del modelo arxqs.

id_m	x_freq	y1_freq	y2_freq	id_fr_resp
10	1.29549	-43.0302	0.861526	820
10	2.1432	-69.3002	0.775557	825
10	9.70396	-231.826	0.202985	840
10	25.8226	-437.686	0.027071	860

Tabla3.2 Segmentos de la tabla frecuency_response.

Tabla3.3 Segmentos de la tabla de correlation_residual_analysis (superior) y cross_correlation_residual_analysis (inferior).

id_m	x1_resid	y1_resid	id_r_a_corr
10	0	1	226
10	8	0.137911	234
10	16	0.114195	242
10	24	0.0675239	250

id_m	x2_resid	y2_resid	id_r_a_cross
10	-24	0.0573294	442
10	-17	-0.0330853	449
10	2	0.225623	468
10	11	0.0277437	477
10	24	-0.0285885	490

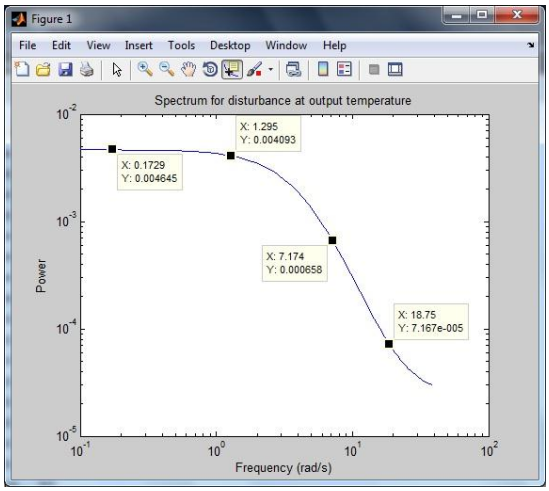


Figura 3.10 Gráfica de espectro de ruido

Tabla 3.4 Segmento de la tabla de noise_spectrum

id_m	x_noi	y_noi	id_noi_spect
10	0.1	0.00465196	685
10	0.172948	0.00464452	691
10	1.29549	0.00409305	712
10	7.17417	0.000658...	729
10	18.7542	0.00007167	740

Para comprobar los vectores se utilizó el comando de MATLAB especificado en la

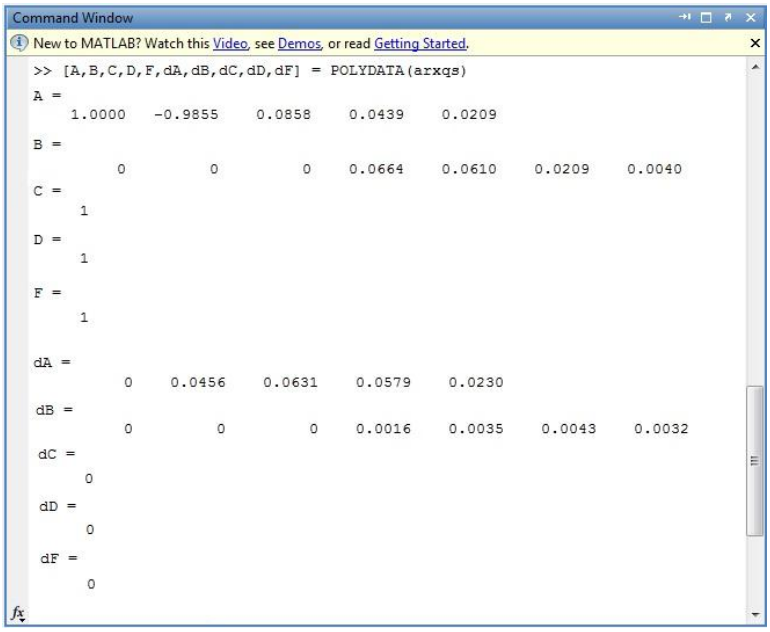


Figura 3.11Ventana de comandos mostrando los resultados de la función polydata.

Tabla 3.5 Segmento de la tabla vector.

id_vec	id_m	v_type	val
271	10	A	1
272	10	A	-0.98546
273	10	A	0.0858069
274	10	A	0.0439481
275	10	A	0.0209412
278	10	B	0
279	10	B	0.0663613
280	10	B	0.0610435
283	10	C	1
284	10	D	1
285	10	F	1
288	10	dA	0.0631193
289	10	dA	0.0579425
290	10	dA	0.0229706
291	10	dB	0
297	10	dB	0.00320501
298	10	dC	0
299	10	dD	0
300	10	dF	0

Los ceros y polos del modelo se comprueban mediante el comando zpkdata de MATLAB como se muestra en la *Figura 3.12*, y los resultados se comparan con la Tabla 3.7.

Nota: El campo z_or_p de la Tabla 3.7, es llenado con ceros, señalando que el registro posee datos de un cero del modelo, o con unos, indicando que el registro posee datos de un polo del modelo.

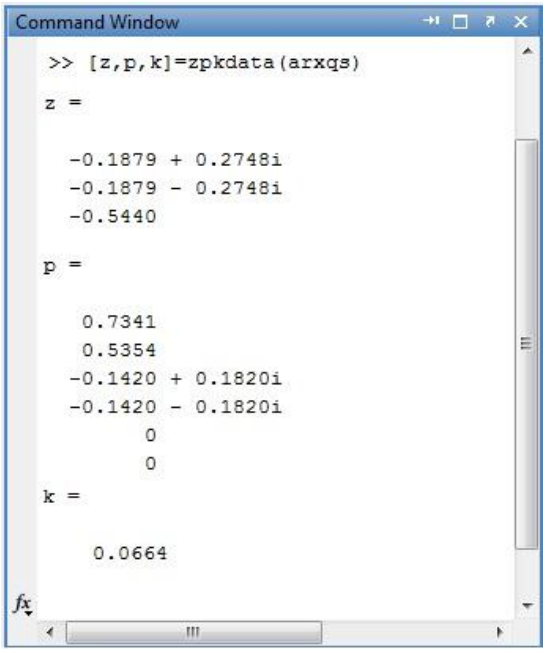


Figura 3.12 resultados de la función zpkdata.

Tabla 3.7 Segmento de la tabla zeros_poles.

id_m	z_or_p	real_part	imag_part	id_zp
10	0	-0.187932	0.274778	82
10	0	-0.187932	-0.274778	83
10	0	-0.544003	0	84
10	1	0.73407	0	85
10	1	0.535351	0	86
10	1	-0.14198	0.182014	87
10	1	-0.14198	-0.182014	88
10	1	0	0	89
10	1	0	0	90

3.2.2 Ejemplo del uso de la función*qidem* y comprobacion de los datos.

Para el uso de la función esta se inserta con diferentes campos en la anterior función, completando así el diagrama EER. Se pretende realizar una serie de pruebas con el objetivo

de demostrar el buen desempeño de la función y la correcta utilización de los criterios implícitos para validar modelos obtenidos mediante la identificación de sistemas.

3.2.2-1 Primera prueba

Para la primera prueba se estimaron ocho modelos, de ellos cuatro presentan estructura ARX y los restantes ARMAX, los datos necesarios para la obtención de los resultados se muestran en la *Figura 3.13.*

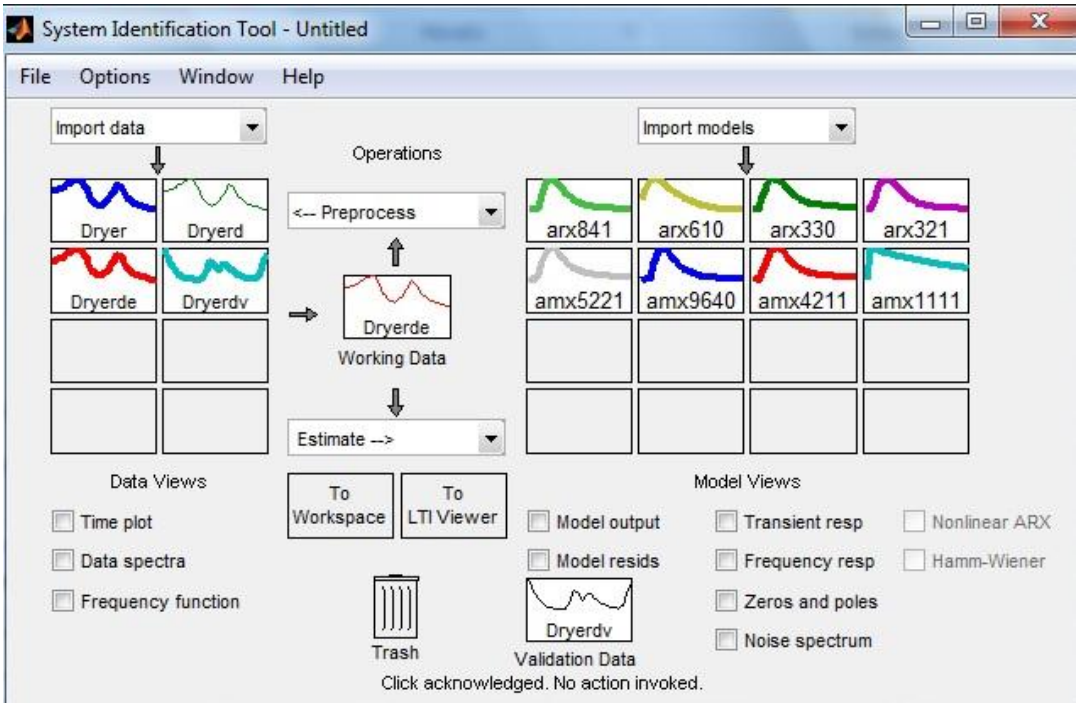


Figura 3.13 Identificación de los modelos para realizar las pruebas de los índices de bondad.

El análisis del mejor de los modelo no puede resumirse a una simple comparación entre estos criterios, debido a que cada uno de ellos mide distintos aspectos. Por tanto, solo fue posible analizar y llegar a conclusiones teniendo en cuenta cada criterio por separado y así determinar cuál de las estructura es la más idónea en cada caso.

Estructura ARX

id_model	id_process	id_loop	date	structure	name	AIC	BIC	CME	R^2
1	proc	loop	2014-05-26	ARX	arx321	0.45928	16.9758	7.0072	0.99156
2	proc	loop	2014-05-26	ARX	arx330	0.47927	20.2991	7.0403	0.99156
3	proc	loop	2014-05-26	ARX	arx610	0.21167	23.3348	3.9756	0.99525
4	proc	loop	2014-05-26	ARX	arx841	0.058456	39.6981	2.1737	0.99747

Figura 3.14 Segmento de la tabla model con la estimación de los índices de bondad (ARX).

Estructura ARMAX

id_model	id_process	id_loop	date	structure	name	AIC	BIC	CME	R^2
1	proc	loop	2014-05-26	ARMAX	amx1111	0.24233	29.9721	107.8028	0.87228
2	proc	loop	2014-05-26	ARMAX	amx4211	1.5223	64.2851	112.59.86	0.87355
3	proc	loop	2014-05-26	ARMAX	amx5221	0.2384	23.3615	64.0006	0.92339
4	proc	loop	2014-05-26	ARMAX	amx9640	0.43799	10.3479	7.1051	0.99136

Figura 3.15 Segmento de la tabla model con la estimación de los índices de bondad (ARMAX).

- **Criterio de información Akaike (AIC)**

El criterio de información de Akaike es una medida de la calidad relativa de un modelo estadístico, para un conjunto dado de datos. Como tal, la AIC proporciona un medio para la selección del modelo. AIC se basa en la entropía de la información: se ofrece una estimación relativa de la información perdida cuando se utiliza un modelo determinado para representar el proceso que realmente genera los datos.

Dado un conjunto de modelos candidatos para los datos, ***el modelo preferido es el que tiene el valor mínimo AIC.***

- **Criterio de información bayesiano (BIC).**

El BIC es calculado para los diferentes modelos como una función de la bondad de ajuste del log Lik, el número de parámetros ajustados (K) y el número total de datos (N).

El modelo con el más bajo valor de BIC es considerado el mejor en explicar los datos con el mínimo número de parámetros.

- **Coeficiente de determinación (R^2).**

Informa sobre la fracción de la varianza total de Y, que es explicada por los modelos y se expresa en porcentaje. Este valor se obtiene a partir de la suma de cuadrados del error (SCE) y de la suma de cuadrados total (SCT).

La SCE corresponde a la suma de cuadrados de las distancias de los puntos desde la curva de mejor ajuste determinada por regresión no lineal, en tanto que la SCT es la suma de cuadrados de las distancias de los puntos desde una línea horizontal correspondiente a la media de todos los valores de Y.

- **Cuadrados medios del residuo (o del error) (CME).**

El CME es una medida que agrupa la variabilidad de aquellos factores que no tiene en cuenta el investigador. Toda vez que el CME corresponde a la varianza residual.

Los modelos seleccionados por su mayor capacidad de ajuste son aquellos que expresan el menor valor en este criterio.

3.2.2-2 Prueba 2

Se desea determinar cuáles son los mejores modelos según los criterios de AIC y R^2 de un total de cuatro modelos que presentan estructura ARMAX.

Para ello se seleccionan e introducen los datos, como en la primera prueba, con la diferencia de que solo se introducen solamente cuatro vectores. En la *Figura 3.16* se muestran los resultados.

id_model	id_process	id_loop	date	structure	name	...	AIC	...	R^2
1	proc	loop	2014-05-26	ARMAX	amx524	...	0.20873	...	0.99545
2	proc	loop	2014-05-26	ARMAX	amx732	...	1.077	...	0.83592
3	proc	loop	2014-05-26	ARMAX	amx642	...	0.15332	...	0.88369
4	proc	loop	2014-05-26	ARMAX	amx551	...	0.258		0.99444

Figura 3.16 Segmento de la tabla model con la estimación de los índices de bondad para la segunda prueba (ARMAX).

Los datos necesarios para realizar la prueba puede ser observada en su totalidad en el Anexo 9.

Se realizó una última prueba para cuatro modelos con estructura ARX, para ser validados utilizando los criterios BIC y CME. Los resultados obtenidos se muestran en la *Figura 3.17*, de forma parcial. En el Anexo 10, se muestra la ventana Ident que se utilizaron para desarrollar la tercera prueba.

id_model	id_process	id_loop	date	structure	name	...	BIC	...	CME
1	proc	loop	2014-05-26	ARX	arx142	...	16.994	...	6.51
2	proc	loop	2014-05-26	ARX	arx653	...	36.6139	...	2.0739
3	proc	loop	2014-05-26	ARX	arx110	...	7.0405	...	5.6288
4	proc	loop	2014-05-26	ARX	arx731	...	33.0808	...	2.1189

Figura 3.17 Segmento de la tabla model con la estimación de los índices de bondad para la segunda prueba (ARX).

3.2.3 Obtención del mejor modelo.

Para la obtención del mejor modelo se trabajó sobre la base de que los criterios no se pueden comparar en conjunto sino hacer un análisis por separado de los mismos siendo el modelo obtenido el que posee de una forma u otra los mejor valores del análisis.

Para ello en la primera prueba se obtuvo el siguiente resultado mostrado en la *Figura 3.18*, del modelo ARX.

id_model	...	structure	...	FIT	AIC	BIC	CME	R^2	na	nb	nc	nd	nf	nk
1	...	ARX	...	73.2805	0.45928	16.9758	7.0072	0.99156	3	3	0	0	0	1
2	...	ARX	...	71.9594	0.47927	20.2991	7.0403	0.99156	3	3	0	0	0	0
3	...	ARX	...	28.4187	0.21167	23.3348	3.9756	0.99525	6	1	0	0	0	0
4	...	ARX	...	89.6037	0.058456	39.6981	2.1737	0.99747	8	4	0	0	0	1

Figura 3.18 Segmento de la tabla model para la primera prueba (ARX).

Esta misma operación se realizó para la primera prueba pero utilizando la estructura ARMAX, ver resultados en la *Figura 3.19.*

id_model	structure	FIT	AIC	BIC	CME	R^2	na	nb	nc	nd	nf	nk
1	ARMAX	24.754	0.24233	29.9721	107.8028	0.87228	1	1	1	0	0	1
2	ARMAX	88.4564	1.5223	64.2851	112.59.86	0.87355	4	2	1	0	0	1
3	ARMAX	89.3579	0.2384	23.3615	64.0006	0.92339	5	2	2	0	0	1
4	ARMAX	89.6639	0.43799	10.3479	7.1051	0.99136	9	6	4	0	0	0

Figura 3.19 Segmento de la tabla model para la primera prueba (ARMAX).

Para la segunda prueba se realiza el mismo procedimiento obteniéndose los siguientes resultados recogidos en la *Figuras 3.20* y *3.21*.

id_model	structure	FIT	BIC	CME	na	nb	nc	nd	nf	nk
1	ARX	87.3982	16.994	6.51	1	4	0	0	0	2
2	ARX	89.3934	36.6139	2.0739	6	5	0	0	0	3
3	ARX	23.3537	7.0405	5.6288	1	1	0	0	0	0
4	ARX	88.9585	33.0808	2.1189	7	3	0	0	0	1

Figura 3.20 Segmento de la tabla model para la segunda prueba (ARX).

id_model	structure	FIT	AIC	R^2	na	nb	nc	nd	nf	nk
1	ARMAX	89.1348	0.20873	0.99545	5	2	4	0	0	1
2	ARMAX	89.4197	1.077	0.83592	7	3	2	0	0	0
3	ARMAX	89.6519	0.15332	0.88369	6	4	2	0	0	1
4	ARMAX	89.5841	0.258	0.99444	5	5	1	0	0	0

Figura 3.21 Segmento de la tabla model para la segunda prueba (ARMAX).

3.3 Análisis económico.

Debido a que se está en presencia de una investigación cuyo mayor aporte está en el campo teórico y no se ha desempeñado con ella un proceso de aplicación práctica estable, el análisis económico se realizará teniendo en cuenta los costos por concepto de adquisición de los softwares y hardware que en ella se utilizaron.

Teniendo en cuenta los anteriormente dicho, los costos de la investigación están dados por la compra de la licencia de *MatLab*, que incluya el *Toolbox "Ident"*, además de

las funciones necesarias para el trabajo. El valor es de 2480 cuc, según el listado de precios consultado (ENERG, 2009). Se incluyen además los costos de la obtención de una PC, monitor, mouse óptico, teclado, UPS, llegando a un costo total de 648 cuc (CIMEX,

2009). En otro orden se puede analizar que el salario de un profesor universitario es de 675.00 cup, por lo que si mensualmente consume el 30 % de su trabajo en el mes en función de la investigación estaríamos hablado por concepto de salario de 236.00 cup (9.44 cuc) que totalizado en los 4 meses de trabajo seria 945.00 cup (37.8 cuc), para finamente tener un costo total de la investigación de 3165.8 en cuc.

Conclusiones

- La función saveintodbnp permite almacenar de forma segura los datos provenientes del proceso de identificación de MATLAB en la base de datos MATLAB_DATA.
- El sistema gestor de base de datos MySQL proporciona confiabilidad, flexibilidad, y buen desempeño en el proceso de almacenamiento de datos.
- La estructura definida para la base de datos permite el rápido almacenamiento y la fácil utilización de los datos por otras aplicaciones para análisis posteriores.
- Mediante criterios de ajuste es posible determinar, de todos los modelos, cuales se adecuan más a la realidad del sistema o proceso. Cada criterio es capaz de medir características específicas del modelo por lo cual no es correcto comparar los modelos a partir de diferentes criterios de bondad.
- La modelación de procesos o sistemas a partir de datos reales es una necesidad de la ingeniería de control moderna ya que permite la prueba de diferentes estrategias de control sin la necesidad de utilizar el proceso o sistema real.

Recomendaciones

- Reestructurar el esquema de la base de datos con el objetivo de incorporar los modelos no lineales, los sistemas MIMO (Multiple Input Multiple Output), así como las demás estructuras no admitidas en la función. Incorporar al código de la función dichos cambios e implementar nuevos algoritmos de inserción para la nueva estructura de la base de datos.
- Si se utilizan versiones de MATLAB y MySQL distintas a las usadas en la programación de la función, actualizar el código de la misma, si es necesario, para eliminar problemas de compatibilidad entre estos softwares y la función.
- Extender los resultados a la obtención de otras estructuras de modelos como ejemplo BJ y OE entre otras.

Referencias bibliográficas

CUADRA, D. 2000. Diseño de base de datos.

GILFILLAN, I. 2003. La Biblia de MySQL.

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

LJUNG, L. 2007. System Identification Toolboox 7. User Guide.

ORACLE, C. 2012. MySQL 5.5 Reference Manual: Including MySQL Cluster NDB 7.2 Reference Guide.

SÁNCHEZ, J. 2004. Diseño conceptual de base de datos.

SCHWARTZ, B., ZAITSEV, P., TKANCHENKO, V., ZAWODNY, J., LENTZ, A. & DEREK, B. 2012. High Performance MySQL: Optimization, Backups, and Replication, O'Reilly Media.

SILBERSCHATZ, A. 2002. Fundamentos de base de datos.

CAMARGO, D. M. O. (2009). Obtención del modelo no paramétrico de un sistema por el método de identificación de respuesta por frecuencia Universidad Pontificia Bolivariana.

OGATA, K. (1998). Modern Control Engineering, Prentice Hall.

PINEDA, J. A. P. (2013). Función para la obtención de datos de identificación desde el MATLAB. Departamento de Automática y Sistemas Computacionales. Santa Clara, Universidad Central "Martha Abreu" de Las Villas.

SIMANCA, J. (2012). CONEM: Herramienta computacional para la adecuada gestión medioambiental de un sector rocoso somero al oeste de La Habana, Cuba.Serie Oceonológica.

SÖDERSTRÖM, T. y. S., P. (1989). System Identification, Prentice Hall.

WHITE, S. 1999. JDBC API tutorial and reference: Universal data access for the Java 2 platform, Chicago, Addison-Wesley Longman Publishing Co., Inc.

VALDIVIA, A. A. 2012. Interfaz para la identificación y ajuste de los controladores

de un simulador de conducción. Tesis de grado, Universidad Central "Marta Abreu" de Las Villas.

SADOSKI, D. 1997. Client/Server Software Arquitecture. An overview, Software Technology Roadmap.

PINTELON, R. & SCHOUKENS, J. 2001. System Identification. A Frequency Domain Approach., New York, IEEE Press.

MIKLES, J. & FIKAR, M. 2007. Process Modelling. Identification and Control, Springer.

MATHWORKS, I. 1998. Database Toolbox. For Use with MATLAB.

MICROSOFT. 2012. Ayuda y soporte técnico [Online]. Available: www.microsoft.com/spain/support/.

MATHWORKS. 2013. Support. Documentation center [Online]. Available: http://www.mathworks.fr/fr/help/index.html.

LÓPEZ, M. E. 2000. Identificación de Sistemas. Aplicación al modelado de un motor de continua. Available: http://www.depeca.uah.es/depeca/repositorio/asignaturas/32328/Tema6.pdf.

AMSTROM, K. J. (1997) Computer Controlled Systems. Theory and Design., Prentice Hall.

Anexo 1: Segmento completo de la tabla *step_response* correspondiente al modelo arxqs.

id_m	x_step	y_step	id_st_resp
10	-0.64	0	397
10	-0.56	0	398
10	-0.48	0	399
10	-0.4	0	400
10	-0.32	0	401
10	-0.24	0	402
10	-0.16	0	403
10	-0.08	0	404
10	0	0	405
10	0.08	0	406
10	0.16	0	407
10	0.24	0.0663613	408
10	0.32	0.192801	409
10	0.4	0.332632	410
10	0.48	0.460664	411
10	0.56	0.56789	412
10	0.64	0.653778	413
10	0.72	0.720661	414
10	0.8	0.771809	415
10	0.88	0.810453	416
10	0.96	0.839409	417
10	1.04	0.86098	418
10	1.12	0.876983	419
10	1.2	0.888821	420
10	1.28	0.897558	421
10	1.36	0.903998	422
10	1.44	0.90874	423
10	1.52	0.912228	424
10	1.6	0.914792	425
10	1.68	0.916677	426
10	1.76	0.918061	427
10	1.84	0.919078	428
10	1.92	0.919825	429
10	2	0.920373	430
10	2.08	0.920776	431
10	2.16	0.921071	432
10	2.24	0.921288	433
10	2.32	0.921448	434
10	2.4	0.921565	435
10	2.48	0.921651	436
10	2.56	0.921714	437
10	2.64	0.92176	438
10	2.72	0.921794	439
10	2.8	0.921819	440

Anexo 2: Segmento completo de la tabla *frequency_response* correspondiente al modelo arxqs.

id_m	x_freq	y1_freq	y2_freq	id_fr_resp
10	0.1	-3.37996	0.921499	793
10	0.104541	-3.53342	0.921463	794
10	0.115615	-3.90759	0.921368	795
10	0.127861	-4.32136	0.921252	796
10	0.141405	-4.77891	0.92111	797
10	0.156383	-5.28486	0.920937	798
10	0.172948	-5.84431	0.920725	799
10	0.191268	-6.4629	0.920467	800
10	0.211528	-7.14685	0.92015	801
10	0.233934	-7.90301	0.919764	802
10	0.258714	-8.73897	0.919292	803
10	0.286118	-9.66308	0.918715	804
10	0.316425	-10.6845	0.918011	805
10	0.349943	-11.8134	0.917151	806
10	0.387011	-13.0608	0.916102	807
10	0.428005	-14.4391	0.914823	808
10	0.473341	-15.9614	0.913264	809
10	0.52348	-17.6426	0.911366	810
10	0.57893	-19.4986	0.909057	811
10	0.640253	-21.5466	0.906251	812
10	0.708073	-23.8057	0.902846	813
10	0.783075	-26.2961	0.898723	814
10	0.866023	-29.0396	0.893738	815
10	0.957757	-32.0595	0.887728	816
10	1	-33.4448	0.884791	817
10	1.05921	-35.3805	0.880503	818
10	1.1714	-39.0285	0.871849	819
10	1.29549	-43.0302	0.861526	820
10	1.43271	-47.413	0.849276	821
10	1.58447	-52.2047	0.834825	822
10	1.75231	-57.4325	0.817895	823
10	1.93792	-63.1227	0.798218	824
10	2.1432	-69.3002	0.775557	825
10	2.37021	-75.9878	0.749729	826
10	2.62128	-83.2057	0.72063	827
10	7.17417	-187.545	0.313071	837
10	7.9341	-201.528	0.273603	838
10	8.77452	-216.271	0.2368	839
10	9.70396	-231.826	0.202985	840
10	10	-236.636	0.193503	841
10	10.7319	-248.258	0.172344	842
10	11.8686	-265.636	0.144937	843
10	13.1258	-284.039	0.120719	844
10	14.5162	-303.542	0.0995604	845
10	14.8171	-307.658	0.0956083	846
10	15.8807	-321.93	0.0831118	847
10	16.2325	-326.56	0.0794301	848
10	17.3522	-341.013	0.0689778	849
10	17.5438	-343.443	0.0673636	850
10	18.7657	-358.666	0.0580905	851
10	19.635	-369.208	0.0524383	852
10	20.6684	-381.437	0.0465737	853
10	20.9955	-385.24	0.0448886	854
10	21.7018	-393.341	0.0415014	855
10	22.0269	-397.021	0.0400498	856
10	23.0166	-408.027	0.0360081	857
10	23.9736	-418.396	0.0325812	858
10	24.9063	-428.249	0.0296354	859
10	25.8226	-437.686	0.027071	860
10	26.7299	-446.796	0.0248121	861
10	27.6355	-455.656	0.0227999	862
10	28.5469	-464.329	0.0209888	863
10	29.4717	-472.867	0.0193429	864
10	30.4179	-481.307	0.0178348	865
10	31.3945	-489.671	0.016445	866
10	31.4289	-489.958	0.0163989	867
10	32.4318	-498.115	0.0151378	868
10	33.321	-504.946	0.0141485	869
10	34.1214	-510.739	0.0133599	870
10	34.6148	-514.131	0.0129225	871
10	34.9869	-516.596	0.0126178	872
10	35.5269	-520.029	0.0122152	873
10	36.1583	-523.831	0.0118059	874
10	36.7555	-527.227	0.0114825	875
10	37.3264	-530.308	0.0112338	876
10	37.8778	-533.156	0.0110519	877
10	38.4156	-535.841	0.0109313	878
10	38.9453	-538.428	0.0108687	879
10	39.0588	-538.978	0.0108626	880

Anexo 3: Segmento completo de la tabla*correlational_residual_analysis*correspondiente al modelo arxqs.

id_m	x1_resid	y1_resid	id_r_a_corr
10	0	1	226
10	1	-0.0410495	227
10	2	0.0413848	228
10	3	0.0861338	229
10	4	0.0353073	230
10	5	0.0654322	231
10	6	0.0441965	232
10	7	0.127626	233
10	8	0.137911	234
10	9	0.0710389	235
10	10	0.111396	236
10	11	0.0900952	237
10	12	0.0476702	238
10	13	0.0948874	239
10	14	0.0614152	240
10	15	0.0771494	241
10	16	0.114195	242
10	17	0.0710563	243
10	18	0.0229921	244
10	19	0.0975038	245
10	20	0.0368445	246
10	21	0.0676392	247
10	22	0.0509111	248
10	23	0.0731751	249
10	24	0.0675239	250

Anexo 4: Segmento completo de la tabla *cross_correlational_residual_analysis* correspondiente al modelo arx541.

id_m	x2_resid	y2_resid	id_r_a_cross
10	-24	0.0573294	442
10	-23	0.0744274	443
10	-22	0.0707812	444
10	-21	0.0720749	445
10	-20	-0.003406	446
10	-19	-0.0022332	447
10	-18	0.0215103	448
10	-17	-0.0330853	449
10	-16	0.0214956	450
10	-15	0.0220678	451
10	-14	0.0599697	452
10	-13	0.0232063	453
10	-12	0.0331217	454
10	-11	-0.00151547	455
10	-10	-0.000806508	456
10	-9	0.0289934	457
10	-8	0.0879972	458
10	-7	0.0904898	459
10	-6	0.0711765	460
10	-5	0.049459	461
10	-4	-0.0436241	462
10	-3	-0.040137	463
10	-2	0.0315155	464
10	-1	-0.0169682	465
10	0	0.0148428	466
10	1	0.101356	467
10	2	0.225623	468
10	3	0.0817419	469
10	4	0.065161	470
10	5	0.107506	471
10	6	0.0860716	472
10	7	0.0274343	473
10	8	0.0145672	474
10	9	-0.00707956	475
10	10	-0.0485061	476
10	11	0.0277437	477
10	12	0.0103535	478
10	13	-0.0154603	479
10	14	-0.038476	480
10	15	0.0152879	481
10	16	0.0236077	482
10	17	0.0646691	483
10	18	0.0562742	484
10	19	0.0118017	485
10	20	0.0115436	486
10	21	-0.0605829	487
10	22	-0.0373253	488
10	23	-0.0496124	489
10	24	-0.0285885	490

Anexo 5: Segmento completo de la tabla *noise_spectrum* correspondiente al modelo arxqs.

id_m	x_noi	y_noi	id_noi_spect
10	0.1	0.00465196	685
10	0.104541	0.00465161	686
10	0.115615	0.0046507	687
10	0.127861	0.00464958	688
10	0.141405	0.00464822	689
10	0.156383	0.00464655	690
10	0.172948	0.00464452	691
10	0.191268	0.00464203	692
10	0.211528	0.00463899	693
10	0.233934	0.00463528	694
10	0.258714	0.00463074	695
10	0.286118	0.0046252	696
10	0.316425	0.00461845	697
10	0.349943	0.00461021	698
10	0.387011	0.00460017	699
10	0.428005	0.00458794	700
10	0.473341	0.00457305	701
10	0.52348	0.00455496	702
10	0.57893	0.00453301	703
10	0.640253	0.0045064	704
10	0.708073	0.00447422	705
10	0.783075	0.0044354	706
10	0.866023	0.00438871	707
10	0.957757	0.00433275	708
10	1	0.00430554	709
10	1.05921	0.00426596	710
10	1.1714	0.00418665	711
10	1.29549	0.00409305	712
10	1.43271	0.00398337	713
10	1.58447	0.00385594	714
10	1.75231	0.00370933	715
10	1.93792	0.00354258	716
10	2.1432	0.00335538	717
10	2.37021	0.00314834	718
10	2.62128	0.00292314	719
10	2.89894	0.00268266	720
10	3.20601	0.00243096	721
10	3.54561	0.00217316	722
10	3.92118	0.00191505	723
10	4.33653	0.00166273	724
10	4.79588	0.001422	725
10	5.30389	0.00119795	726
10	5.8657	0.000994...	727
10	6.48703	0.000814...	728
10	7.17417	0.000658...	729
10	7.9341	0.000525...	730
10	8.77452	0.000416...	731
10	9.70396	0.000327...	732
10	10	0.000304...	733
10	10.7319	0.000256...	734
10	11.8686	0.000200...	735
10	13.1258	0.000157...	736
10	14.5162	0.000123...	737
10	15.8807	0.000101...	738
10	31.0026	0.000034...	752
10	32.036	0.000033...	753
10	33.0694	0.000032...	754
10	34.1028	0.000031...	755
10	35.1362	0.000031...	756
10	36.1697	0.000030...	757
10	37.2031	0.000030...	758
10	38.2365	0.000030...	759
10	39.0588	0.000030...	760

Anexo 6 : Segmento completo de la tabla *vector* correspondiente al modelo arxqs.

id_vec	id_m	v_type	val
271	10	A	1
272	10	A	-0.98546
273	10	A	0.0858069
274	10	A	0.0439481
275	10	A	0.0209412
276	10	B	0
277	10	B	0
278	10	B	0
279	10	B	0.0663613
280	10	B	0.0610435
281	10	B	0.0209232
282	10	B	0.00400073
283	10	C	1
284	10	D	1
285	10	F	1
286	10	dA	0
287	10	dA	0.0455727
288	10	dA	0.0631193
289	10	dA	0.0579425
290	10	dA	0.0229706
291	10	dB	0
292	10	dB	0
293	10	dB	0
294	10	dB	0.00159166
295	10	dB	0.00346845
296	10	dB	0.00432919
297	10	dB	0.00320501
298	10	dC	0
299	10	dD	0
300	10	dF	0

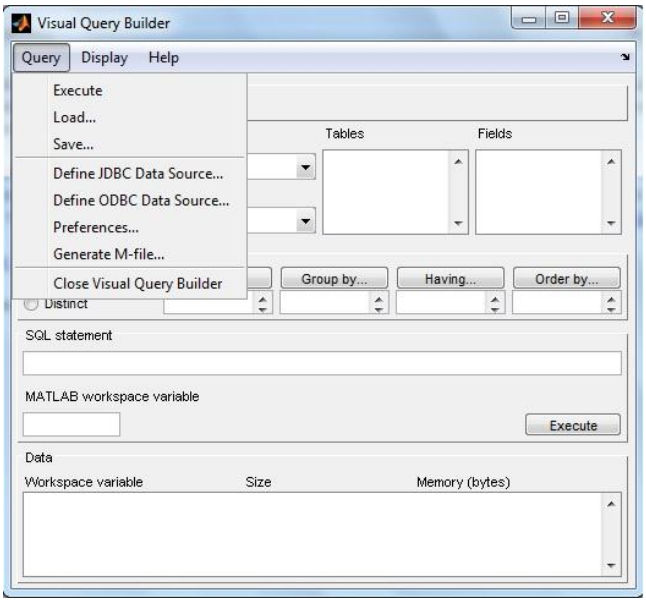
Anexo 7: Configuración de fuentes de datos para su uso con controladores ODBC.

En este anexo se describe cómo configurar un origen de datos cuya base de datos se encuentra en un PC con el sistema operativo Windows, o en otro sistema al que está conectado en red del PC, para su uso con un controlador ODBC. Estas instrucciones utilizan el ODBC MySQL Connector/ODBC 5.1. Si tiene una configuración diferente, es posible que tenga que modificar estas instrucciones:

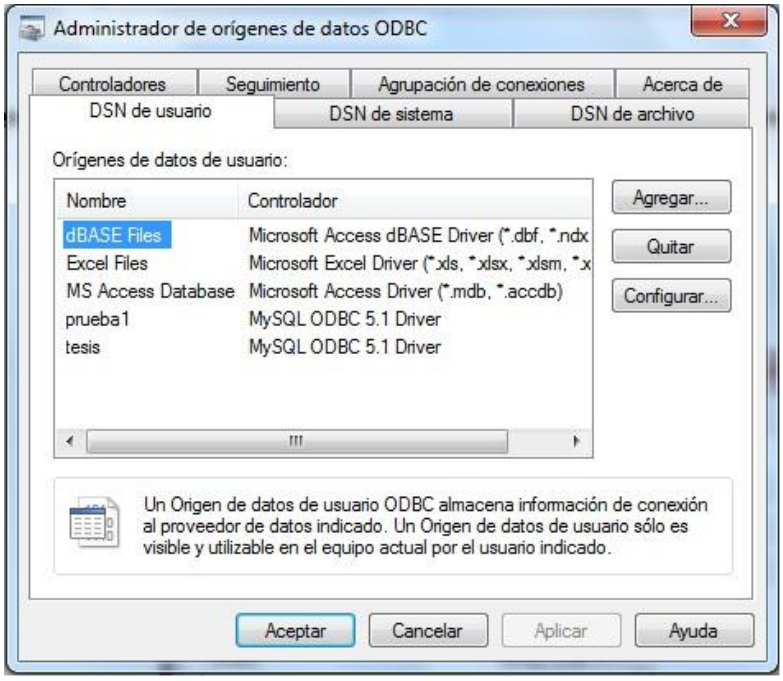
1. Cerrar bases de datos abiertas en el programa de base de datos.
2. Abra el cuadro de diálogo Windows Data Source para definir el origen de datos ODBC

a) Inicie el constructor de consultas visual (Visual QueryBuilder) introduciendo el comando querybuilder en el prompt de MATLAB.

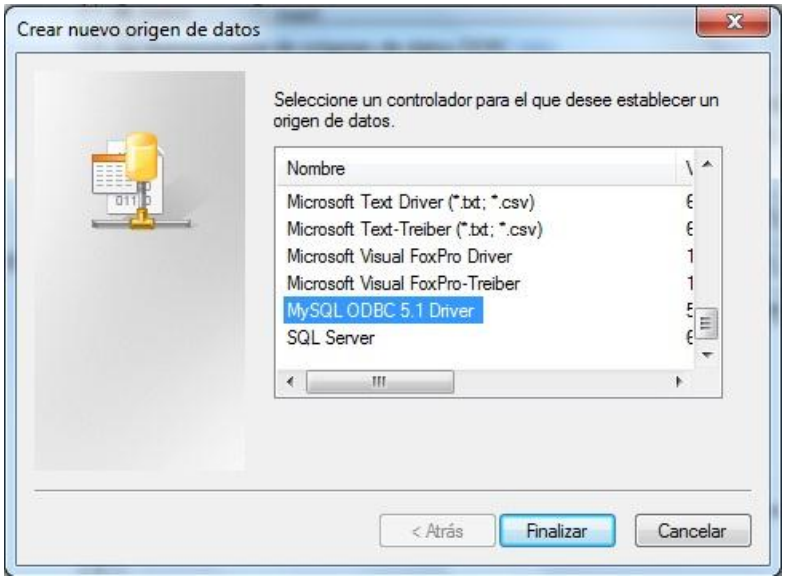
b) En Visual Query Builder, seleccione **Query > Define ODBC Data Source.**



3. Aparecerá el cuadro de diálogo Administrador de orígenes de datos ODBC (ODBC Data Source Administrator), enumerando las fuentes de datos existentes.
4. Pulsando en "Agregar" desde "DSN de sistema" o "DSN de usuario".
5. Seleccione MS Access Database y haga clic en Add.



6. El asistente para crear un nuevo origen de datos nos mostrará los controladores, entre ellos seleccionaremos el instalado anteriormente "MySQL ODBC 5.1 Driver" y pulsaremos "Finalizar":

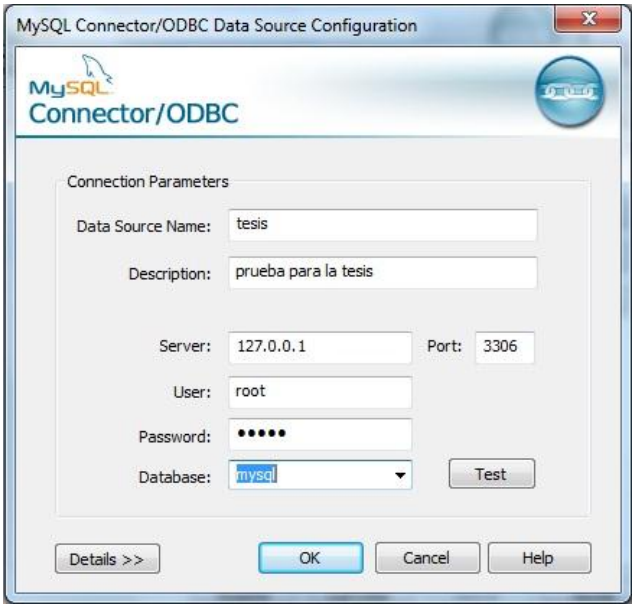


En la ventana de MySQL Connector/ODBC Data Source Configuration introduciremos los siguientes datos para crear el nuevo origen de datos ODBC:

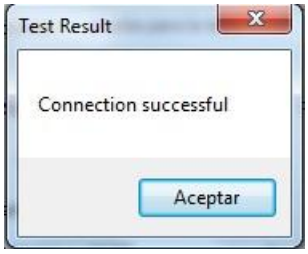
- **Data SourceName**: nombre identificativo del origen de datos ODBC, por ejemplo "bdajpdsoft".
- **Description**: breve descripción para saber identificar el origen de datos cuando tenemos varios, por ejemplo "Conexión BD MySQL desde Access".

- **TCP/IP Server**: introduciremos la dirección IP, la URL o el nombre de red (hostname) del servidor con el motor de base de datos MySQL Server.
- **Port**: indicaremos el puerto establecido en el servidor de MySQL Server para las conexiones externas, por defecto 3306.
- **User**: nombre de usuario de MySQL Server para realizar la conexión a la base de datos.
- **Password**: contraseña del usuario de MySQL Server anterior.
- **Database**: una vez introducidos los datos anteriores, si el usuario indicado tiene permisos suficientes, en el desplegable de "Database" podremos seleccionar la base de datos (catálogo) al que nos conectaremos.

Tras introducir los datos de conexión al servidor MySQL Server podremos hacer un test (prueba de conexión) pulsando en el botón "Test":



Si los datos son correctos y el servidor de MySQL Server está disponible y la conexión se establece correctamente mostrará el siguiente mensaje:



Con el texto: Connection successful.

Anexo 8: Configuración de fuentes de datos para su uso con los controladores JDBC.

Para configurar orígenes de datos para su uso con los controladores JDBC:

1. Busque el nombre del archivo que contiene los controladores JDBC.

2. Especifique la ubicación del archivo de controladores en *the* MATLAB *Sun Java* classpath añadiendo la ruta de este archivo al archivo:

matlabroot/toolbox/local/classpath.txt.

Por ejemplo, use la siguiente sintaxis para agregar una referencia a un archivo de controladores MySQL® JDBC archivo: *D:/mysql/mysql-connector-java-3.0-bin.jar, a classpath.txt:*

D:/mysql/mysql-connector-java-3.0-bin.jar

Reinicie la sesión del software MATLAB antes de acceder a la base de datos. Si el archivo de controladores no se encuentra en classpath.txt indica, que no aparecen errores, pero MATLAB no establece la conexión con la base de datos.

3. Haga lo siguiente para configurar los drivers JDBC para usar con Visual QueryBuilder:

a. Inicie el constructor de consultas visual (*Visual QueryBuilder*) introduciendo el comando *querybuilder* en el *prompt* de MATLAB.

b. En *Visual QueryBuilder*, seleccione **Query> Define JDBC Data Source.** El cuadro de diálogo *Define JDBC Data Sources* aparece.

En la siguiente tabla se describen los campos que se utilizan para definir el origen de datos JDBC.

Nombre del campo en el cuadro de diálogo *Define JDBC Data Sources*	Descripción
Nombre	El nombre que se asigna a la fuente de datos. En algunas bases de datos, el nombre debe coincidir exactamente con el nombre de la base de datos como se reconoce en la máquina en que se ejecuta.
Controlador	El nombre del controlador JDBC (a veces conocido como la clase que implementa el controlador de Java SQL para la base de datos).
URL	El URL JDBC, de la forma `jdbc:subprotocol:subname`. `Subprotocol` es un tipo de base de datos, por ejemplo, la base de datos de Oracle ® Corporation Oracle. `subname` puede contener otra información utilizada por el controlador, tal como la ubicación de la base de datos y / o un número de puerto. Puede tomar la forma `//hostname:port/databasename`

c. En el cuadro de diálogo *Define JDBC data sources*, haga clic en **Create New File.**

d. Aparece el nuevo cuadro de diálogo *Specify new JDBC data source MAT-file*. Utilice este cuadro de diálogo para crear un *MATLAB MAT-file* que guarda la información de origen de datos especificado para las futuras sesiones de *Visual QueryBuilder*.

Navegue a la carpeta donde desea colocar el archivo MAT, especifique un nombre para él, que incluye una extensión .mat, y haga clic **Save**.

e. En el cuadro de diálogo *Define JDBC Data Sources*, ingrese la información de la fuente de datos JDBC en los campos: nombre, controlador y URL, como se describe en la tabla del paso 3b.

f. Pruebe la conexión haciendo clic en **Test**.

Si su base de datos requiere un nombre de usuario y contraseña, aparecerá un cuadro de diálogo que le pide que proporcione ellos. Introduzca los valores en estos campos y haga clic en **Ok**.

Aparecerá un cuadro de diálogo de confirmación indica que la conexión de base de datos se ha realizado correctamente.

g. En el cuadro de diálogo *Define JDBC Data Sources*, haga clic en Agregar / actualizar. El origen de datos aparece ahora en la lista Origen de datos.

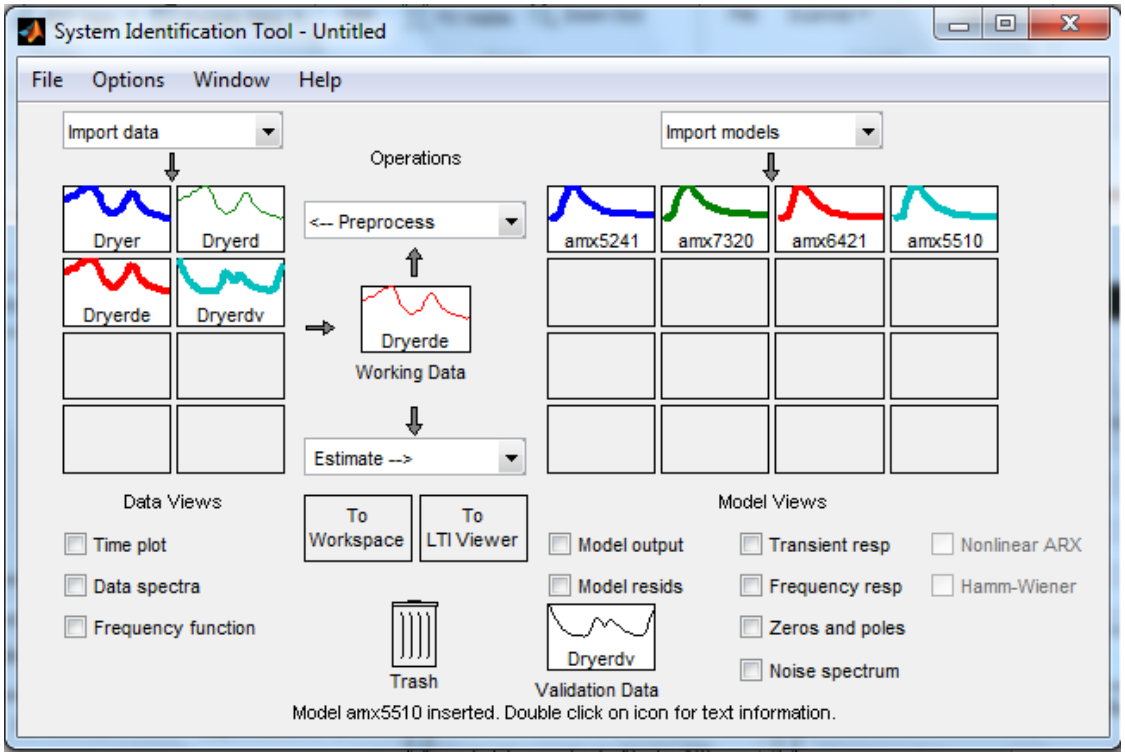
h. Para agregar más fuentes de datos, repita los pasos del c al g para cada nueva fuente de datos. Usted puede agregar más fuentes de datos en el futuro mediante la edición del archivo-MAT que contiene la información del origen de datos.

Nota: Puede crear otro MAT-file para añadir nuevas fuentes de datos, pero Visual QueryBuilder sólo puede acceder a fuentes de datos de un MAT-archivo a la vez.

Incluye múltiples fuentes de datos en un único archivo MAT -para facilitar el acceso.

i. Haga clic en Aceptar para cerrar el cuadro de diálogo Definir datos JDBC fuentes. La fuente de datos recién agregado aparece ahora en la lista Origen de datos VQB. La fuente de datos JDBC definido sólo es válida para la sesión actual del *software* MATLAB.

Anexo 9: Datos de la segunda prueba, estructura ARMAX.



Anexo 10: Datos de la segunda prueba, estructura ARX.