PLATAFORMA PARA OBTENER LA RESPUESTA EN FRECUENCIA

MANUEL RICARDO VARGAS AVILA
FELIPE ANDRES PINEDA PRADA
Estudiantes de Ingeniería Electrónica

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS Y ADMINISTRACIÓN
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
SEPTIEMBRE 2010

PLATAFORMA PARA OBTENER LA RESPUESTA EN FRECUENCIA

MANUEL RICARDO VARGAS AVILA
FELIPE ANDRES PINEDA PRADA
Estudiantes de Ingeniería Electrónica

Tesis de grado presentada como requisito para optar el título de Ingeniero Electrónico

DIRECTOR DEL PROYECTO:

HÉCTOR RAMIRO PÉREZ RODRÍGUEZ
Ph.D. Ingeniería Mecánica

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS Y ADMINISTRACIÓN
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
SEPTIEMBRE 2010

Nota de aceptación

_____
_____
_____
_____
_____
_____

_____

Firma del jurado

_____

Firma del jurado

Bucaramanga, Septiembre de 2010

Dedico a este proyecto a mi familia y amigos por su apoyo incondicional, a ampliar mis conocimientos, y especialmente a Dios por haberme dado la oportunidad de hacer este proyecto tan maravilloso, y darme la fuerza para realizar mis sueños, y poder seguir luchando y seguir creciendo como persona y como ingeniero.

Gracias Dios, gracias a ti estoy aquí.

MANUEL RICARDO VARGAS AVILA

Quiero dedicar la realización de este proyecto de forma muy especial a toda mi familia, quienes siempre me brindaron su apoyo incondicional, dándome fuerzas en aquellos momentos difíciles y celebrando en los momentos de felicidad. De igual forma, a todos mis amigos y compañeros que de una u otra manera hicieron que mis conocimientos crecieran gracias a cada uno de sus aportes. A Dios por brindarme salud y haberme dado la oportunidad y las fuerzas necesarias para cumplir cada una de mis metas, permitiéndome seguir progresando como persona y como profesional.

FELIPE ANDRES PINEDA PRADA

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

A nuestro Director de Tesis: Ph.D. HECTOR RAMIRO PEREZ RODRIGUEZ por su gran apoyo incondicional, por su carisma y sabiduría para poder sacar este proyecto adelante.

A nuestro profesor PhD. Omar Pinzón Ardila por sus aportes materiales. Igualmente por la disposición de sus conocimientos teórico-prácticos en las actividades que conllevaron el buen desarrollo del proyecto.

A nuestro profesor Edgar Barrios, por su apoyo en las inquietudes que tuvimos durante el proyecto.

Y finalmente a todas las personas que estuvieron de alguna manera vinculadas en la elaboración y desarrollo de este proyecto. Gracias a todos.

LISTA DE FIGURAS

TABLAS

ALGORITMOS

ANEXOS
ANEXO 1

ANEXO 2

ANEXO 3

ANEXO 4

ANEXO 5

ANEXO 6

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: PLATAFORMA PARA LA IDENTIFICACION DE SISTEMAS BASADO EN LA RESPUESTA EN FRECUENCIA

AUTOR(ES): Felipe Andrés Pineda Prada
Manuel Ricardo Vargas Avila

FACULTAD: Facultad de Ingeniería Electrónica

DIRECTOR(A): Héctor Ramiro Pérez Rodríguez

En este proyecto se desarrolló e implementó una plataforma de hardware y software que permite para un sistema, planta o proceso obtener: (1) la respuesta a un escalón, (2) el modelo por ocho métodos de identificación que utilizan como entrada la respuesta en el tiempo (3) simulación del modelo identificado a entradas sinusoidales (4) identificación el sistema con base en la respuesta en frecuencia (5) la respuesta a un escalón del modelo encontrado (6) el espectro en frecuencia de la señales sinusoidales adquiridas. Como aplicación, se emplearon los utilitarios de la plataforma a un motor DC del módulo del laboratorio de control de la Universidad Pontificia Bolivariana Seccional Bucaramanga. La plataforma está conformada por un computador con el software MATLAB® y de la tarjeta de adquisición de datos USB 6008 de la National Instruments. El módulo de práctica además de la plataforma utiliza el conjunto de motor DC y la tarjeta de interface desarrollados por Feedback®. De MATLAB® se utilizan las herramientas GUIDE, Data Acquisition Toolbox (Librería de Adquisición de Datos) y Control System Toolbox. Si la tarjeta USB 6008 se emplea para alimentar una planta, el ancho de banda de la plataforma máximo es de 75 Hz (por el criterio de Nyquist) y si sólo se emplea para recibir los datos provenientes de la planta el ancho de banda es de 5 Khz (por el criterio de Nyquist).

PALABRAS CLAVES: Identificación de Sistemas, Matlab, Feedback, GUIDE, Tarjeta USB 6008.

GENERAL SUMMARY OF WORK OF DEGREE

TITLE: PLATFORM FOR THE SYSTEM IDENTIFICATION BASED ON THE FREQUENCY RESPONSE

AUTHORS: Manuel Vargas Ricardo Ávila
Felipe Andrés Pineda Prada

FACULTY: Electronic Engineering Faculty

DIRECTOR: Héctor Ramiro Pèrez Rodríguez

This project was developed and implemented a hardware and software platform that allows for a system, plant or process to obtain: (1) the frequency response,(2) the step response, (3) the model of eight methods for identification using as input the time response and (4) the simulation of step response applied to the models inferred from the response in time and frequency response. From the frequency response is obtained by the Bode plot and frequency spectrum resulting from applying the Fourier transform of the response of a plant when a sinusoidal signal applied. As a practical application, utilities were applied to the platform module DC motor control laboratory of the Universidad Pontificia Bolivariana Bucaramanga Sectional. The platform consists of a computer with the software MATLAB ® card and USB data acquisition from National Instruments 6008. The practical module of the platform also uses the DC motor assembly and the interface card developed by Feedback ®.MATLAB ® tools are used GUIDE, Data Acquisition Toolbox (Library Data Acquisition) and Control System Toolbox. The performance of the platform depends on the characteristics of the data acquisition card in this project was the USB card from National Instruments 6008. The sampling rate for input channels is 10 kHz maximum sampling rate for output channels is 150 Hz output channels are used to send the signal to the plant (eg a DC motor) . Therefore, if the USB card 6008 is used to feed a plant, the bandwidth of the highest platform is 75 Hz (the Nyquist criterion) and if only used to receive data from the plant widthbandwidth is 5 kHz (the Nyquist criterion).

KEYWORDS: Systems of Identification, Matlab, Feedback, GUIDE, USB 6008 CARD.

INTRODUCCION

Equipos para identificar un sistema con base en la respuesta en frecuencia existen en el mercado mundial desde hace varios años. Sin embargo, la universidad no dispone de esos equipos. Es importante que la universidad disponga del mismo y lo utilice tanto en el pregrado como en el postgrado para que estudiantes y profesores dispongan de una herramienta que ayude a afianzar los conocimientos de respuesta en frecuencia con sus diagramas de Bode y se vea la aplicabilidad de los mismos. Tampoco hay disponibilidad de Herramientas que identifiquen una planta por varios métodos. La plataforma conjuga ambos aspectos, se obtiene la respuesta en frecuencia y la respuesta en el tiempo, y con base en ambos tipos de respuesta se infiere el modelo de la planta.

La plataforma implementada sirve de punto de partida en diversidad de proyectos de control, ya que prácticamente todos ellos requieren la respuesta en frecuencia y/o la respuesta en el tiempo y el modelo inferido a partir de las respuestas.

La plataforma le ayuda a la comunidad universitaria a entender y visualizar la aplicabilidad de los conceptos teóricos. Es frecuente que los estudiantes pregunten, ¿estando en la vida laboral, como puedo obtener la función de transferencia de un equipo que escasamente lo veo trabajar y no sé que tiene por dentro? Bueno, la plataforma permite conocer teórica y prácticamente la función de transferencia de una planta o proceso. Es interesante notar que en muchos casos la única información disponible de las plantas o procesos son las entradas que se le aplican a la planta o proceso y la respuesta a esas entradas aplicadas, por lo que es importante tener a la mano un dispositivo que interpretando las entradas y la respuesta me identifique el sistema.

PROPOSITO

La plataforma será usada por estudiantes de pregrado y postgrado. Este informe debe servir como manual de instrucción para que el lector pueda entender y operar un analizador de señales dinámicas. Este analizador de señal dinámica fue específicamente desarrollado para funcionar con la tarjeta NATIONAL INSTRUMENTS USB 6008, con herramientas básicas disponibles en MATLAB, entre ellas la TOOLBOX de adquisición de datos, funciones como LINSPACE, LSIM, PADE, STEP, BODE entre otros. Todos son recursos valiosos para investigar características de la planta y su identificación a través de la respuesta en frecuencia.

DESCRIPCIÓN GENERAL DE USO DE LA PLATAFORMA

Primero se le aplica al sistema una entrada escalón, y se adquiere su respuesta con el fin de determinar el tiempo de establecimiento, es decir, el tiempo en el que la respuesta del sistema está en estado estable. Como la señal adquirida posee ruido blanco, esta debe ser filtrada por un filtro de suavizado llamado SMOOTH para eliminarlo.

Conocida la respuesta al escalón (step en inglés) se determina su comportamiento y se identifica si la respuesta es amortiguada, sub-amortiguada o sobre-amortiguada. Con esta información se define el modelo dinámico que más se ajusta al sistema: primer orden más tiempo muerto, segundo orden sobre-amortiguado más tiempo muerto o segundo sub-amortiguado más tiempo muerto. Se selecciona el método de identificación y se obtiene la función de transferencia que representa al sistema.

Se simula la respuesta en el tiempo del modelo identificado a entradas sinusoidales utilizando la función LSIM de MATLAB. Las señales de entrada y salida de la simulación se visualizan respectivamente.

Luego, se somete la planta, para este proyecto el motor DC, a señales de entrada sinusoidales a través de la tarjeta de la National Instruments USB 6008 y con la misma tarjeta se captura la respuesta del sistema. Las señales de entrada y salida se grafican y se visualizan en la plataforma. Este proceso se repite a diferentes frecuencias.

A medida que la frecuencia va aumentado, en el caso del motor, se observa una señal de baja frecuencia en la señal adquirida.

Para identificar esta frecuencia se utiliza el análisis espectral que arroja la TRANSFORMADA DE FOURIER DISCRETA. Con el análisis espectral de la señal, se encuentran las magnitudes y las fases de los componentes de la señal a diferentes frecuencias. Para limpiar la señal de salida, se emplea un filtro pasa alta digital. El espectro que se obtiene con la Transformada rápida de Fourier (Fast Fourier Transform (fft)) es visualizado y el usuario puede interactuar con él, con el fin de que decida si la señal necesita la ayuda del filtro pasa alta o no para la frecuencia que se esté trabajando.

Teniendo la señal totalmente sin ruido, se identifica su amplitud y el desfase con respecto a la entrada. El mismo proceso se hace con la señal de entrada, con la diferencia que esta señal sólo la variamos en frecuencia, por lo que ya conocemos su amplitud.

La amplitud y el desfase de la señales son visualizadas en la plataforma para que el usuario pueda ver en tiempo real la variación de las mismas.

Las variables de amplitud, desfase y frecuencia se van guardando para poder construir la respuesta en frecuencia representada en un diagrama de Bode. La magnitud del diagrama de Bode se obtiene dividiendo la magnitud de la señal de salida sobre la magnitud de la señal de entrada. La fase se obtiene midiendo el desfase de la señal de salida con respecto a la señal de entrada.

A cada frecuencia de la señal de entrada le corresponde una magnitud en decibeles y una fase en grados.

El diagrama de Bode es una de las representaciones de la respuesta en frecuencia, el cual se obtiene, graficando en el eje de las x la frecuencia en una escala logarítmica y en dos gráficas diferentes, la magnitud en decibeles y la fase en grados.

La respuesta en frecuencia real se compara con las respuestas típicas para deducir la función de transferencia de la planta o proceso que se esté analizando. Identificado el modelo en forma de función de transferencia se puede realizar la estrategia de control que se desee.

Todos los elementos teóricos requeridos para este proyecto se conocen. Este proyecto toma esos criterios y los monta en una plataforma que guía y permite obtener la función de transferencia.

La plataforma de hardware y software queda disponible en los laboratorios de control y se puede utilizar en prácticas de pregrado y/o postgrado.

Como productos finales además de la plataforma a implementar se tiene el libro del proyecto y un manual de uso de la plataforma. En ellos se incluye el uso de la interfaz MATLAB® y de la tarjeta de adquisición de datos, la obtención de la respuesta en frecuencia y resultados experimentales. En el manual se explica detalladamente el uso de la plataforma GUIDE. Otro producto del proyecto es un artículo que se somete para publicación en la revista Puente de la Dirección General de Investigaciones (DGI) de la Universidad Pontificia Bolivariana Seccional Bucaramanga.

MOTIVACION PARA IMPLEMENTAR LA PLATAFORMA.

El deseo y la necesidad de la comunidad universitaria de la Universidad Pontificia Bolivariana Seccional Bucaramanga de tener en sus laboratorios de control una plataforma para obtener la respuesta en frecuencia y en el tiempo e identificar sistemas partiendo de los dos tipos de respuesta se llevó a la realización de éste proyecto.

La identificación del modelo de un sistema es un tema de interés creciente para la comunidad universitaria por su practicidad en la industria, conocimiento que si es adquirido apalanca a los nuevos profesionales. El interés por disponer de una plataforma de este tipo ha surgido en los estudiantes que toman la optativa de identificación y sintonización de controladores PID y que ven en esta plataforma una herramienta clave para el afianzamiento de sus conocimientos. Esta plataforma sirve como punto de partida para la realización de nuevas investigaciones por parte de estudiantes de pregrado y postgrado.

DESCRIPCIÓN DEL CONTENIDO DEL DOCUMENTO

El capitulo 1 describe el módulo del motor. El capitulo 2 describe la tarjeta de interface desarrollada por FEEDBACK. El capitulo 3 define que es un diagrama de bode y que lo compone. El capitulo 4 describe que es la correlación lineal y como se utiliza. El capitulo 5 define los tipo de filtros que se utilizaron en la plataforma. En el capítulo 6 define que parámetros se debe tener para poder obtener la respuesta en frecuencia. En el capítulo 7 describe los métodos que se utilizaron para identificar el sistema a través de la respuesta en el tiempo. El capitulo 8 describe los componentes que componen la plataforma GUIDE diseñada. El capitulo 9 define unos puntos críticos para poder adquirir señales a través de MATLAB. Ya si el usuario quiere profundizar en algún capitulo descrito, puede proceder a mirar los anexos.

OBJETIVOS

OBJETIVO GENERAL

El objetivo general es el desarrollo de un dispositivo que permite obtener la respuesta en frecuencia de un sistema representada en el diagrama de Bode y con base en la respuesta en frecuencia soportar al usuario en la obtención del modelo representada en la función de transferencia.

OBJETIVOS ESPECÍFICOS

1. Realizar transferencia de tecnología relacionada con la respuesta en frecuencia a la comunidad universitaria. Los productos de este objetivo son la publicación de un libro disponible para la comunidad universitaria con la teoría de la respuesta en frecuencia, su obtención, su interpretación y resultados experimentales y seminario o curso corto que se dictara al público en general.

2. Realizar transferencia de tecnología relacionada con la comunicación de MATLAB® con el medio. El producto de este objetivo es la publicación en el libro de tesis de la forma de realizar la comunicación entre MATLAB® y el medio.

3. Obtener la respuesta en frecuencia del motor de Feedback® existente en el laboratorio de control de la Universidad Pontifica Bolivariana Seccional Bucaramanga. El producto es el reporte experimental paso a paso en la obtención de la respuesta en frecuencia y el modelo correspondiente.

4. Implementar métodos de identificación de: sistemas de primer orden más tiempo muerto, sistemas sobre-amortiguados de segundo orden más tiempo muerto, y sistemas de segundo orden sub-amortiguados para obtener el modelo a través de la respuesta en el tiempo.

5. Difundir la producción intelectual del grupo de investigación Automatización Instrumentación y Control (AIC).

1. MODULO DEL MOTOR

A través de Matlab se envía señales hacia el motor y el motor genera señales que son adquiridas por la tarjeta de adquisición, los cuales son llevados a Matlab a través del adaptador NI-DAQ o NI-DAQmx.

Este sistema consta de las siguientes partes (Vea figura 1):

- Motor de corriente continua, que acciona una determinada carga en forma de disco. Este disco se puede frenar de forma manual mediante un freno magnético. En este eje se aloja un tacogenerador, o sensor de velocidad, que proporciona una medida de la velocidad del eje.

- Engranaje de relación 32:1 que transmite el movimiento del eje motor a una carga.

- Carga cuya posición se desea controlar. Para ello, se dispone de un sensor óptico de posición (*encoder*) que proporciona una medida de la posición.

Así pues, el sistema consta de un motor accionado por su tensión de alimentación que acciona, a través de un engranaje, el eje de carga en el que se sitúan los sensores de posición y velocidad.

Figura 1. Módulo del Motor

2. UNIDAD PARA SISTEMAS DE CONTROL

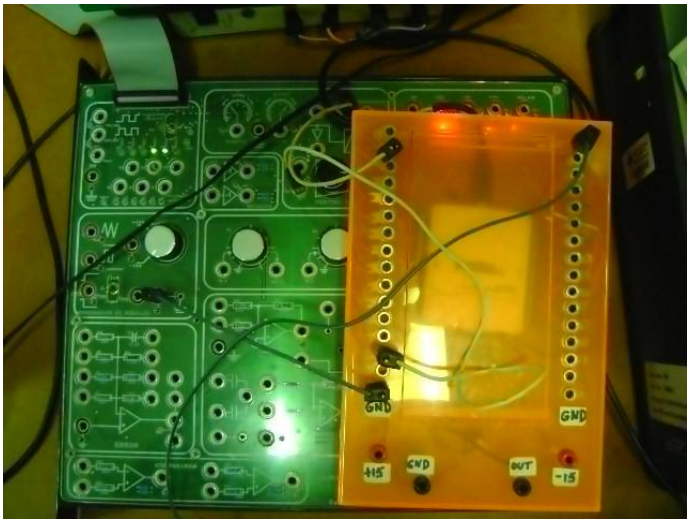


Figura 2. Tarjeta de interface

Este módulo facilita la conexión tanto interna como externamente entre el PC y la tarjeta de adquisición USB 6008. Es un módulo de gran ayuda debido a la gran variedad de componentes y pines en los cuales se pueden realizar diferentes acciones. Entre sus componentes se puede encontrar desde pines para aterrizar las señales, potenciómetros para determinar el giro del motor, pines de amplificación, entre otros.

3. DIAGRAMA DE BODE

El diagrama de Bode es un tipo de representación gráfica de funciones complejas (en nuestro caso, funciones de transferencia), el cual nos muestra el comportamiento de la magnitud y la fase de una planta cuando se excita a entradas sinusoidales. Como los datos de magnitud y desfase tomados de la planta varían entre sí varios órdenes, se procede a graficar el diagrama de bode en una escala logarítmica. En un diagrama de bode el eje x representa la frecuencia en rad/seg, y el eje y representa la magnitud y la fase (Vea figura 3).

La magnitud del diagrama de bode representa la relación de la amplitud de la sinusoide de salida y de entrada de la planta, y la fase representa la diferencia entre el ángulo de fase de la sinusoide de salida y el ángulo de fase de la sinusoide de entrada.

La magnitud del diagrama de bode se encuentra de la siguiente manera:

$$A(w) = 20 * \log\left(\frac{yo}{uo}\right)$$

Donde *yo* es la amplitud de la señal de salida y *uo* es la amplitud de la señal de entrada.

La fase del diagrama de bode se encuentra de la siguiente manera:

$$Y(w) = (-w * (t2 - t1) * 180°)/\pi$$

Donde *w* es la frecuencia en rad/seg, y la diferencia de *t2* y *t1* representa el retardo de la señal de salida respecto a la señal de entrada. Para más información diríjase a [14].

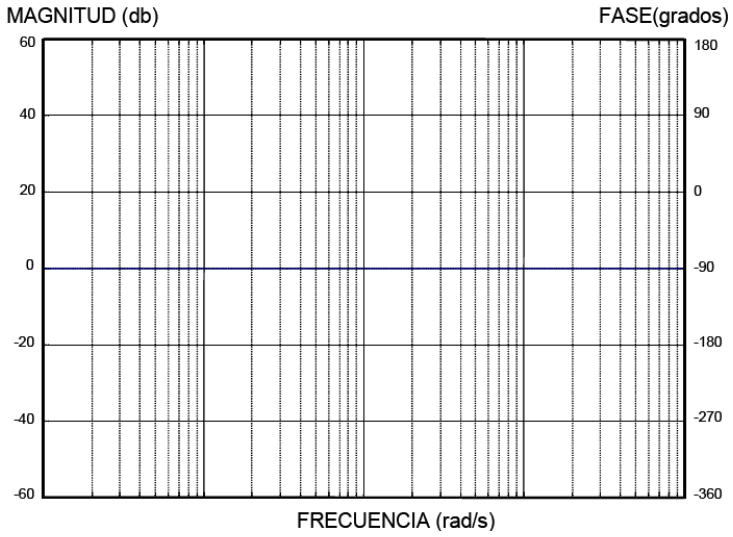


Figura 3. Diagrama de bode

4. CORRELACION LINEAL

El coeficiente de correlación lineal mide el grado de intensidad de la relación entre las variables. Este tipo de correlación se aplica solo cuando la relación que puede existir entre las variables es lineal (es decir, si representáramos en un gráfico los pares de valores de las dos variables la nube de puntos se aproximaría a una recta).

El valor del coeficiente de correlación (r) puede variar entre -1 y 1, dependiendo del coeficiente se puede decir que:

Si $r > 0$, la correlación lineal es positiva (si sube el valor de una variable sube el de la otra). La correlación es tanto más fuerte cuanto más se aproxime a 1.

Si $r < 0$, la correlación lineal es negativa (si sube el valor de una variable disminuye el de la otra). La correlación negativa es tanto más fuerte cuanto más se aproxime a -1.

Si $r = 0$, no existe correlación lineal entre las variables. Aunque podría existir otro tipo de correlación (parabólica, exponencial, etc.)

La función que representa el coeficiente de correlación lineal en MATLAB es: $r = corrcoef(x, y)$, donde *x* y *y* son los vectores de datos.

El coeficiente de correlación se aplica en la plataforma GUIDE para poder saber cuál de los métodos identificados con la respuesta en el tiempo a un STEP se aproximan mas a la respuesta real, y así poder compararlo con la respuesta al STEP del modelo obtenido de la respuesta en frecuencia.
El método de primer orden con el coeficiente de correlación más cercano a 1 es el método de Alfaro. Al graficar el error entre la respuesta de la planta y el método Alfaro tienen una relación lineal (ver figura 4), y su coeficiente de correlación es del 0.99987. Para más información diríjase a [11]

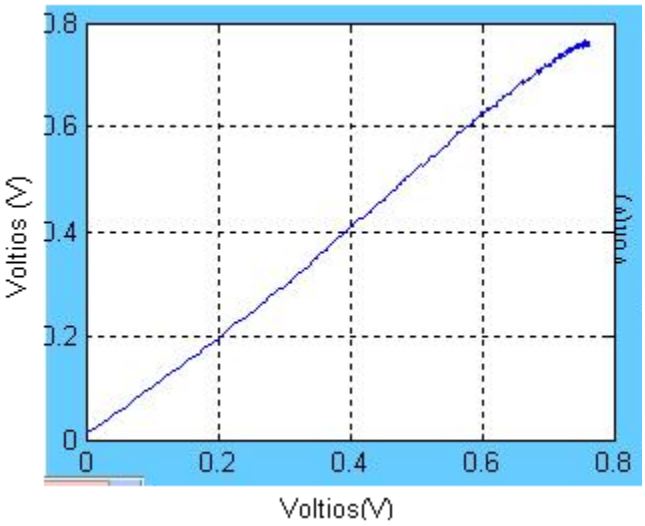


Figura 4. Correlación lineal (Alfaro vs planta)

5. FILTROS

En la plataforma se utilizaron 2 tipos de filtros digitales, cada uno con un fin diferente, uno para suavizar y el otro para eliminar frecuencias.

- **FILTRO SMOOTH:**

Es un filtro de suavizado, su propósito uso es el de suavizar la señal reduciendo el ruido blanco al azar mientras se mantiene la respuesta al escalón más aguda. Esto se hizo especificando el método de filtro de media móvil ('moving'), el cual es un filtro en el dominio del tiempo usado en acciones tales como: suavizado, supresión DC, formado de ondulación, etc.

Este es configurado de la siguiente manera:

$$yy = smooth(y)$$
$$yy = smooth(y, span)$$
$$yy = smooth(y, method)$$
$$yy = smooth(y, span, method)$$

El span establece la duración de la media móvil a palmo. Span debe ser impar.

- **FILTRO FILTFILT:**

Es un filtro digital que reduce al mínimo la puesta en marcha y termina transitorios, haciendo coincidir las condiciones iníciales. Primero filtra el vector *x*, y su respuesta la rota y le vuelve a aplicar el mismo filtro. La respuesta final evita la distorsión de fase propia de los filtros IIR.

Este es configurado de la siguiente manera:

$$y = filtfilt(b, a, x)$$

Donde *b* y *a* representan el numerador y el denominador del tipo de filtro que se aplicar a la señal, ya sea pasa baja, pasa alta o pasa banda y *x* es la señal.
Para generar los coeficientes del filtro que se va aplicar, se utiliza el **filtro Butter**, el cual se caracteriza por una respuesta de magnitud que es máximamente plana en la banda de paso y monótona en general. Estos filtros pueden ser utilizados como filtro pasa bajo, pasa alta y pasa banda, dependiendo de la necesidad de aplicación.

Se configura de la siguiente manera.

$$[y1, y2] = butter(n, wn)$$
$$[y1, y2] = butter(n, wn,' flype')$$

Donde *n* es el orden del filtro, *wn* es la frecuecia de corte normalizada y *flype* es el tipo de filtro 'high' , 'low' , o 'stop'. Para el ‘stop’ se especifican las 2 frecuencias de la siguiente manera remplazando Wn, [Wn1 Wn2]. Y1 y y2 son los coeficiente del filtro.

En la plataforma se usa un filtro pasa alta debido a que se tiene que eliminar una baja frecuencia que no pertenece a la señal. Para poder eliminarla se debe tener cuidado al escoger el valor de Wn debido a que este valor es normalizado, este procedimiento se hace de la siguiente manera:

$$wn = \frac{wc}{\frac{frecuencia\ de\ la\ señal}{2}}$$

$$Donde\ wc = \frac{frecuencia\ a\ eliminar}{frecuencia\ de\ la\ señal}$$

Ahora sólo es reemplazar Wn en el filtro. Para mas información diríjase a [8]

6. PARAMETROS PARA OBTENER LA RESPUESTA EN FRECUENCIA

Para construir la respuesta en frecuencia del sistema, se debe aplicar a la entrada una señal sinusoidal. A la salida del sistema en estado estable, se obtiene otra onda sinusoidal de la misma frecuencia que la señal de entrada, la cual a medida que se varía la frecuencia de la señal de entrada, puede tener una amplitud diferente y estar desfasada en el tiempo respecto a la entrada (ver figura 5).

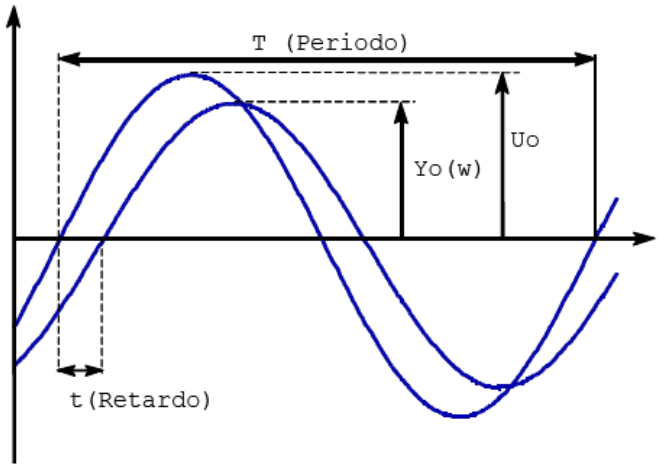


Figura 5. Parámetros para obtener la respuesta en frecuencia

Una vez aplicada la señal de entrada se obtiene la señal de salida. Se debe tener en cuenta que: $\boldsymbol{Uo}$ representa la amplitud de la señal de entrada, $\boldsymbol{Yo(w)}$ es la amplitud de la señal de salida, $\boldsymbol{T}$ es el periodo de la señal de entrada y $\boldsymbol{t}$ representa el retardo de la señal de salida respecto a la señal de entrada. Estos datos se toman variando la frecuencia de la señal de entrada. Con estos valores se obtiene la magnitud [db], la fase [grados] del sistema y la frecuencia en rad/seg .(Vea Tabla 1).

$\boldsymbol{Periodo}$ $T(s)$	$\boldsymbol{Frec(hz)}$ $f=\frac{1}{T}$	$\boldsymbol{frec(\frac{rad}{seg})}$ $w=2*\pi*f$	**Entrada** Uo	**Salida** $Yo(w)$	**Magnitud** $A(w)=20*\log(\frac{yo(w)}{u0})$	$\boldsymbol{retardo}$ t	$\boldsymbol{Fase}$ $\varphi(w)=\frac{-w*t*180^{\circ}}{\pi}$

Tabla 1. Parámetros a obtener la respuesta en frecuencia

7. IDENTIFICACION DE SISTEMAS (RESPUESTA EN EL TIEMPO)

Para la identificacion del sistema con la respuesta en el tiempo, se utilizan diversos metodos dependiendo de la respuesta al escalon del sistema real, ya sea sobreamortiguada o sub-amortiguada. Para mas informacion dirijase a [7]

Los modelos dinamicos utilizados son:

- Primer orden mas tiempo muerto

$$Gp(s)=\frac{kpe^{-tm*s}}{\tau s+1}$$

- Segundo orden sobre-amortiguado más tiempo muerto

$$Gp(s)=\frac{kpe^{-tm*s}}{(\tau 1s+1)(\tau 2s+1)}$$

- Segundo orden sub-amortiguado

$$Gp(s) = \frac{kp\ wn^2}{s^2 + 2\,\xi\ wn\ s + wn^2}$$

➢ METODOS DE IDENTIFICACION (METODOS DE LAZO ABIERTO)

Los modelos de segundo orden o mayor pueden ser aproximados mediante un modelo de primer orden más tiempo muerto. La gran mayoría de los procesos son identificados de esta manera, pero algunos de ellos necesitan modelarse por un sistema de segundo orden más tiempo muerto. Para aproximar a un modelo de primer orden más tiempo muerto, se utiliza el método de dos puntos. Para aproximar a un modelo de segundo orden más tiempo muerto se utiliza el método de tres puntos.

- **METODO DE DOS PUNTOS:**

Los métodos que requieren el trazo de la recta tangente a la curva no siempre son fáciles de hacer, debido a que algunos pueden contener ruido y será difícil obtener el punto máximo de la pendiente. Esto lleva a que la recta que se trazo no sea la ideal, afectando el tiempo muerto y la constante de tiempo del modelo. Debido a esto se creó otro método utilizando 2 puntos de la curva de reacción (Vea Figura 6 y Tabla 2). A partir de este método se puede modelar el sistema a un modelo de primer orden más tiempo muerto o un modelo de polo doble más tiempo muerto.

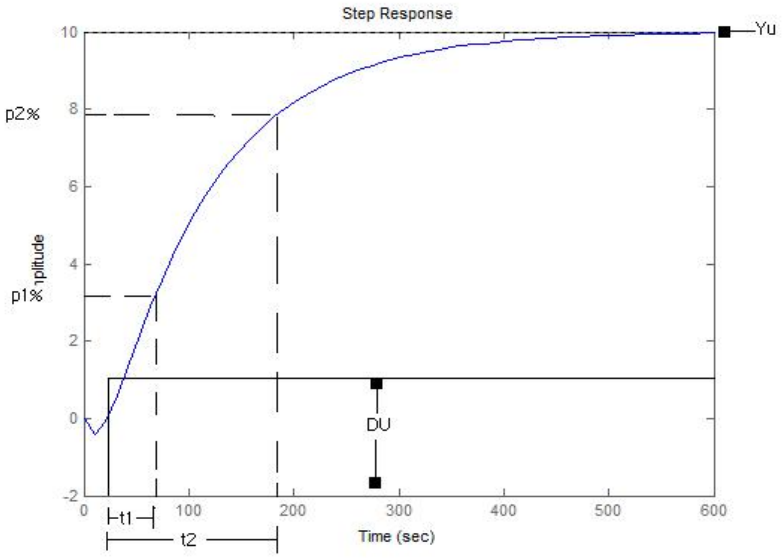


Figura 6. Método de dos puntos

Método	% p1(t1)	% p2(t2)	A	b
Alfaro	25	75	0.91	1.26
Broida	28	40	0.55	2.80
Ho	35	85	0.67	1.29
Chen y yang	33	67	0.14	1.54
Smith	28	63	0.15	1.50
Viteckova	33	70	1.24	1.49

Tabla 2. Constantes de métodos de primer orden más tiempo muerto

- **MODELO DE PRIMER ORDEN MAS TIEMPO MUERTO**

Para identificar el modelo existen diversos métodos, entre ellos: **Alfaro, broida, ho, chen y yang, Smith y viteckova**, cada uno maneja unos porcentajes diferentes con respecto al valor final y poseen 2 constantes a y b. Para encontrar el τ, kp y el tm usan las siguientes formulas:

$$kp = \frac{Yu}{\mathrm{D}u}$$

$$\tau = a * (t2 - t1)$$

$$tm = b * t1 + (1 - b) * t2$$

- **METODO DE TRES PUNTOS**

Al igual que los métodos de dos puntos, este método no necesita del trazado de la recta tangente a la curva. Este método toma 3 puntos de la respuesta. Son el $15\%, 45\%\ y\ el\ 75\%$ con respecto al valor final (ver figura 7). A partir de este método se puede modelar el sistema a un modelo de segundo orden sobre amortiguado más tiempo muerto.

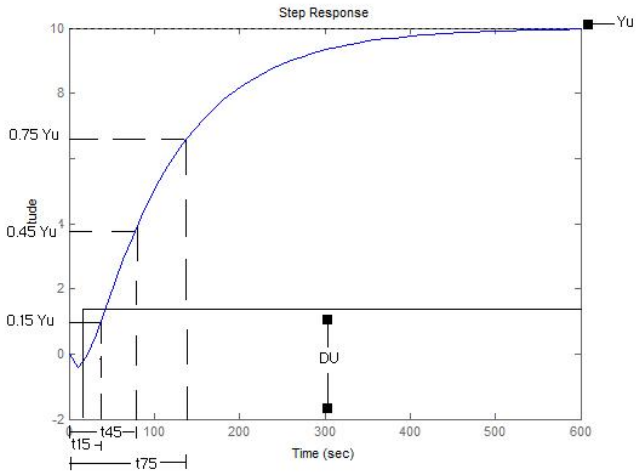


Figura 7. Método de tres puntos

- **MODELO DE SEGUNDO ORDEN SOBRE AMORTIGUADO MAS TIEMPO MUERTO**

Para identificar el modelo existe un método llamado método de **Stark,** el cual teniendo los 3 puntos identifica el modelo.
Para encontrar $kp, tm, \tau 1\ y\ \tau 2$, se utilizan las siguientes formulas:

$$t1 = 15\%, t2 = 45\%, t3 = 75\%$$

$$kp = \frac{Yu}{Du}$$

$$x = \frac{t2 - t1}{t3 - t1}$$

$$\xi = \frac{0.0805 - 5.547(0.475 - x)^2}{x - 0.356}$$

$$f_2(\xi) = 2.6\,\xi - 0.60 \quad para\ \xi > 1$$

$$wn = \frac{f_2(\xi)}{t3 - t1}$$

$$f_3(\xi) = 0.922\ (1.66)^{\xi}$$

$$tm = t2 - \frac{f_3(\xi)}{wn}$$

$$\tau 1 = \frac{\xi + \sqrt{\xi^2 - 1}}{wn} \quad para\ \xi > 1$$

$$\tau 2 = \frac{\xi - \sqrt{\xi^2 - 1}}{wn} \quad para\ \xi > 1$$

- **MODELO DE SISTEMAS SUB-AMORTIGUADOS DE SEGUNDO ORDEN O MAYOR**

Cuando la respuesta del sistema a un cambio de tipo escalón en la señal de entrada, tiene un comportamiento sub-amortiguado, es erróneo utilizar modelos de primer orden o segundo orden con polos reales para su identificación. Es necesario utilizar un modelo que tenga polos complejos conjugados.

- **METODO DE IDENTIFICACION BASADO EN EL SOBRE PASO MAXIMO**

Permite identificar sistemas sub-amortiguados de orden superior o igual a dos (Vea figura 8).

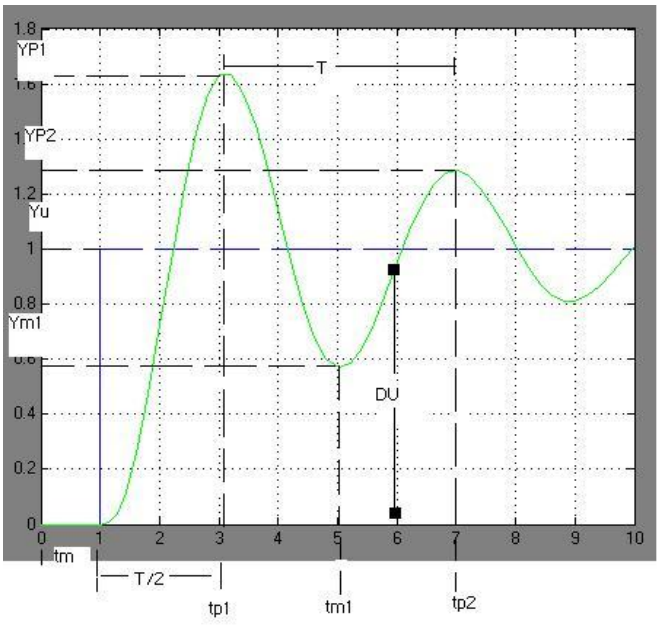


Figura 8. Mètodo de identificacion basado en el sobrepaso maximo

$$Kp = \frac{(y_u - y_{0)}}{\Delta u}$$

$$\delta = \frac{y_{p1} - y_u}{y_u - y_0}$$

$$\xi = \sqrt{\frac{ln^2\delta}{\pi^2 + ln^2\delta}}$$

$$T = 2(t_{m1} - t_{p1})$$

$$w_{\boldsymbol{n}} = \frac{2\pi}{T\sqrt{1-\xi^2}}$$

$$t_m = t_{p1} - \frac{T}{2}$$

8. PLATAFORMA GUIDE

El propósito del proyecto es identificar un sistema a través de la respuesta en frecuencia y en tiempo, por lo que se creó un espacio de trabajo con la GUI de Matlab. Desde allí se envían y se reciben datos, asimismo se puede dibujar el diagrama de bode.

Igualmente el sistema es identificado a través de las formas de identificación existentes, éstas varían dependiendo de la respuesta del sistema al aplicar un Step. De igual forma, se encuentra el diagrama de bode y se compara con el bode del sistema identificado.

La plataforma esta divida en 2 secciones (Vea figura 9), en la obtención del modelo por la respuesta en frecuencia y la respuesta en el tiempo. Para más información diríjase a anexo [ ]

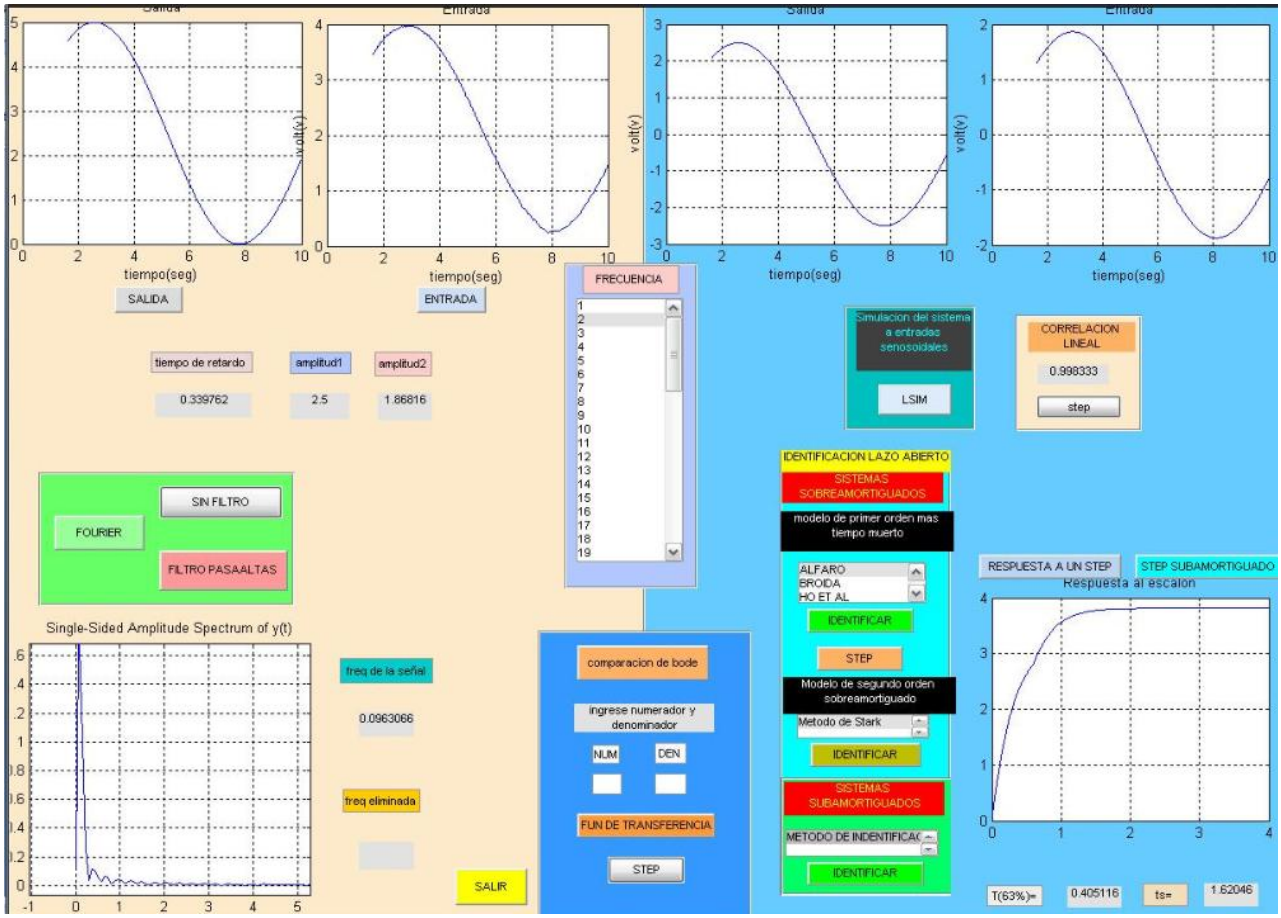


Figura 9. Plataforma completa

8.1 RESPUESTA A UN STEP

Para conocer cuál es el tiempo de establecimiento del motor, se envía un escalón por el puerto digital hacia él y se adquiere su respuesta al STEP (Vea Figura 10). A la respuesta adquirida, se emplea el filtro de suavizado SMOOTH, debido a su contenido ruidoso. Al tener filtrada la respuesta, se encuentra el valor del tiempo cuando la respuesta alcanza el 63% de su valor final, obteniendo con éste el tiempo de establecimiento. El parámetro se guarda en la variable xy, y el tiempo de establecimiento en TS2. El valor de TS2, se busca en el vector tiempo para ver en que localidad se encuentra. Su código es representado en el algoritmo 1.

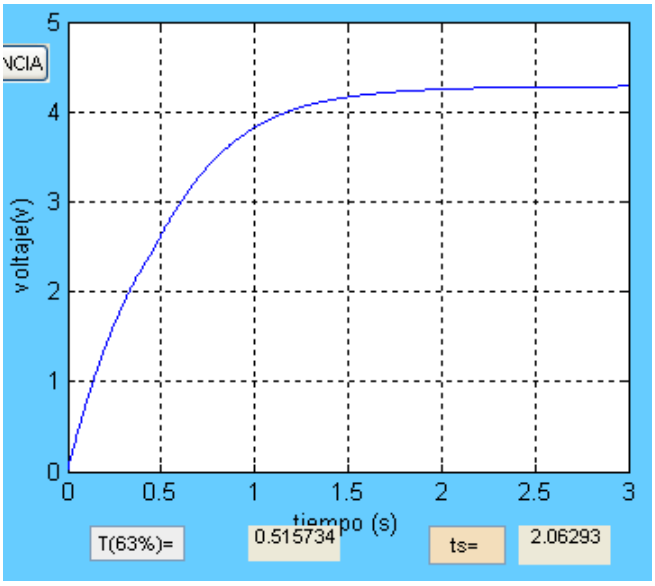
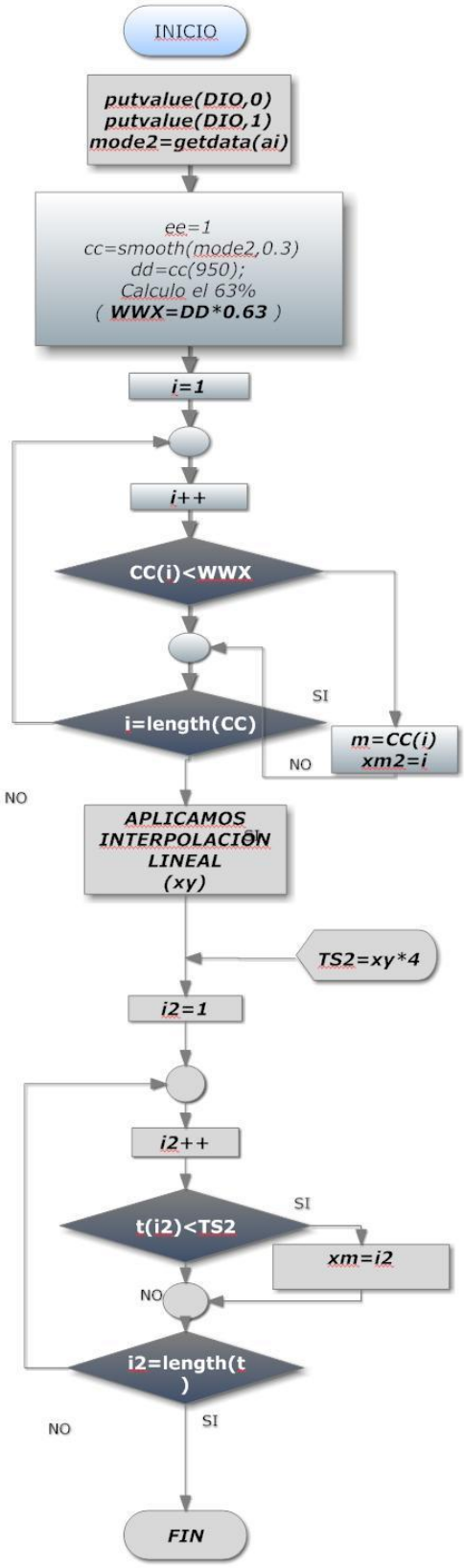


Figura 10. Respuesta a un Step

Yu(max)=4.2v

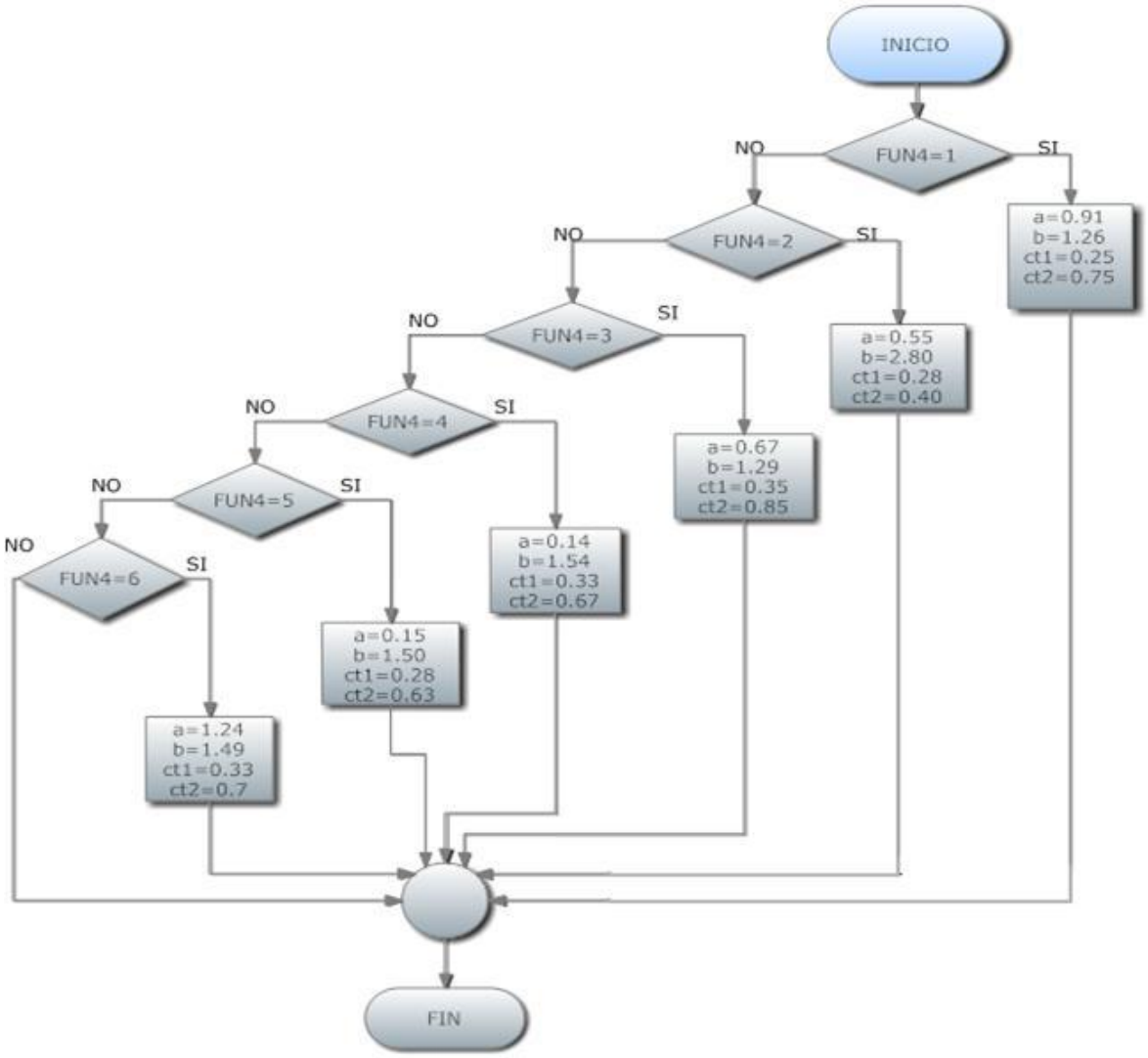


Algoritmo 1. Respuesta a un Step

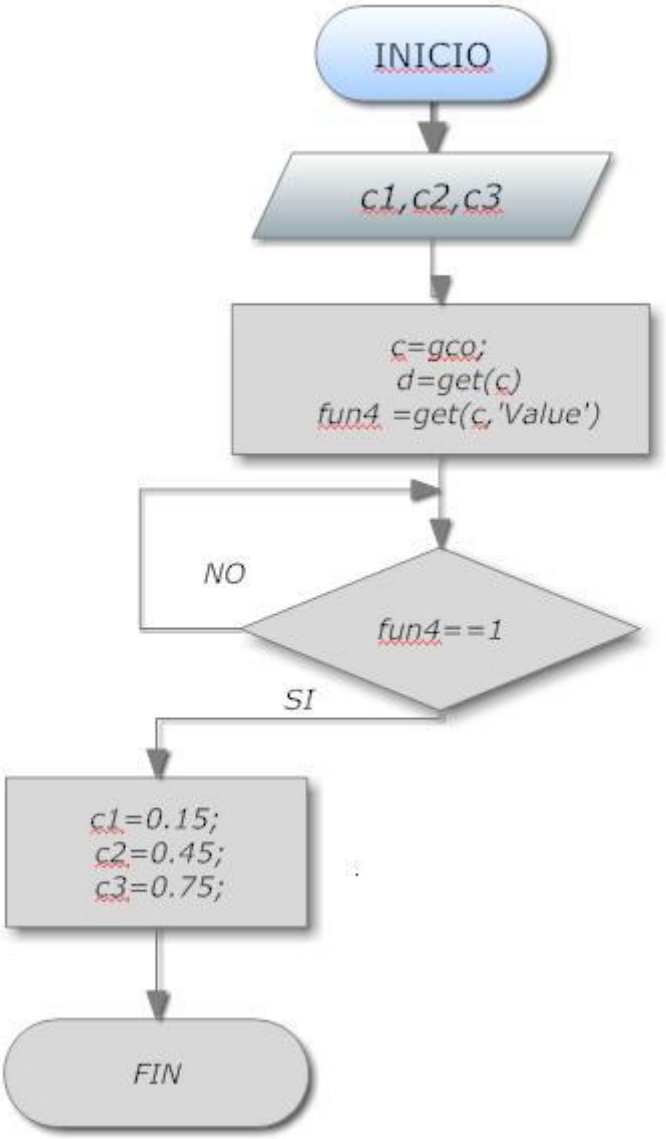
8.2 IDENTIFICACION DEL MODELO (LAZO ABIERTO)

Se identifica el sistema, a través de algunos métodos conocidos. Esta sección esta divida en 3 tipos de modelos, primer orden más tiempo muerto, segundo orden más tiempo muerto y 2 orden sub-amortiguado. Se escoge el modelo a aproximar y se selecciona el método con el cual se va a modelar el sistema, y se encuentran las constantes que construyen la función de transferencia dependiendo del modelo escogido. Se le aplica un escalón, se obtiene su respuesta y se compara con la respuesta al STEP de la planta real.

Su código es representado en el algoritmo 2 y 3.



Algoritmo 2. Métodos primer orden



Algoritmo 3. Método segundo orden

8.3 IDENTIFICAR 1, 2, 3

Al tener como finalidad el identificar el sistema, se debe obtener el valor del polo y asimismo el valor del tiempo muerto. Existiendo T (lo que esT), es comparado con el hallado anteriomente, con el objetivo de saber si el valor del polo obtenido es cercano al sistema identificado. Su código es representado en el algoritmo 4, 5 y 6.

- **PRIMER ORDEN OBTENIDO**

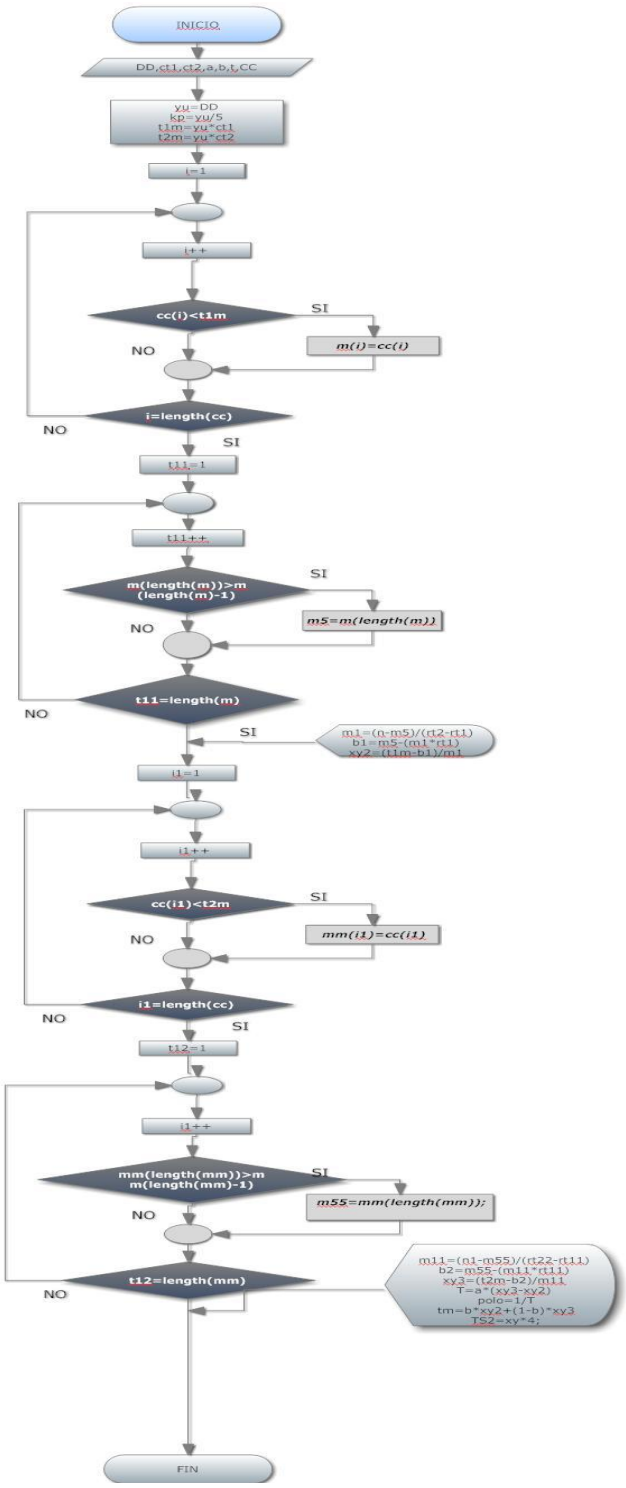
$$\frac{0.7634}{0.3456\ s + 1}$$

- **SEGUNDO ORDEN OBTENIDO**

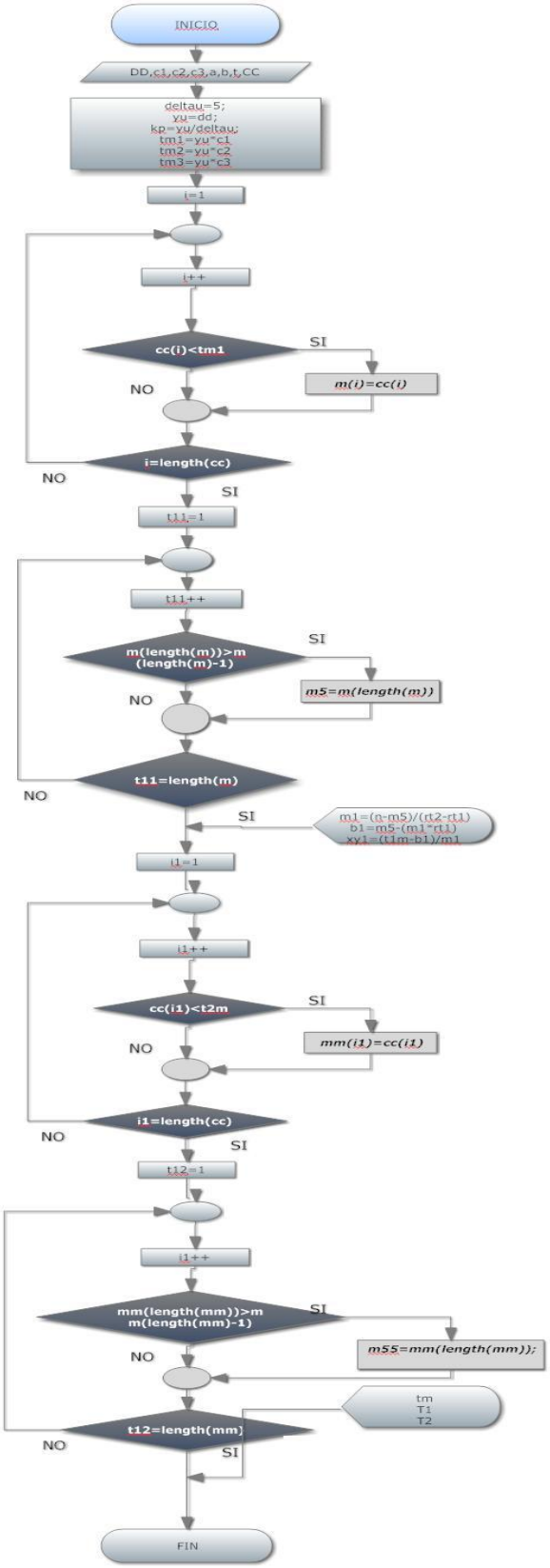
$$\frac{0.7634}{0.01449\ s^2 + 0.3758\ s + 1}$$

- **SUB-AMORTIGUADO**

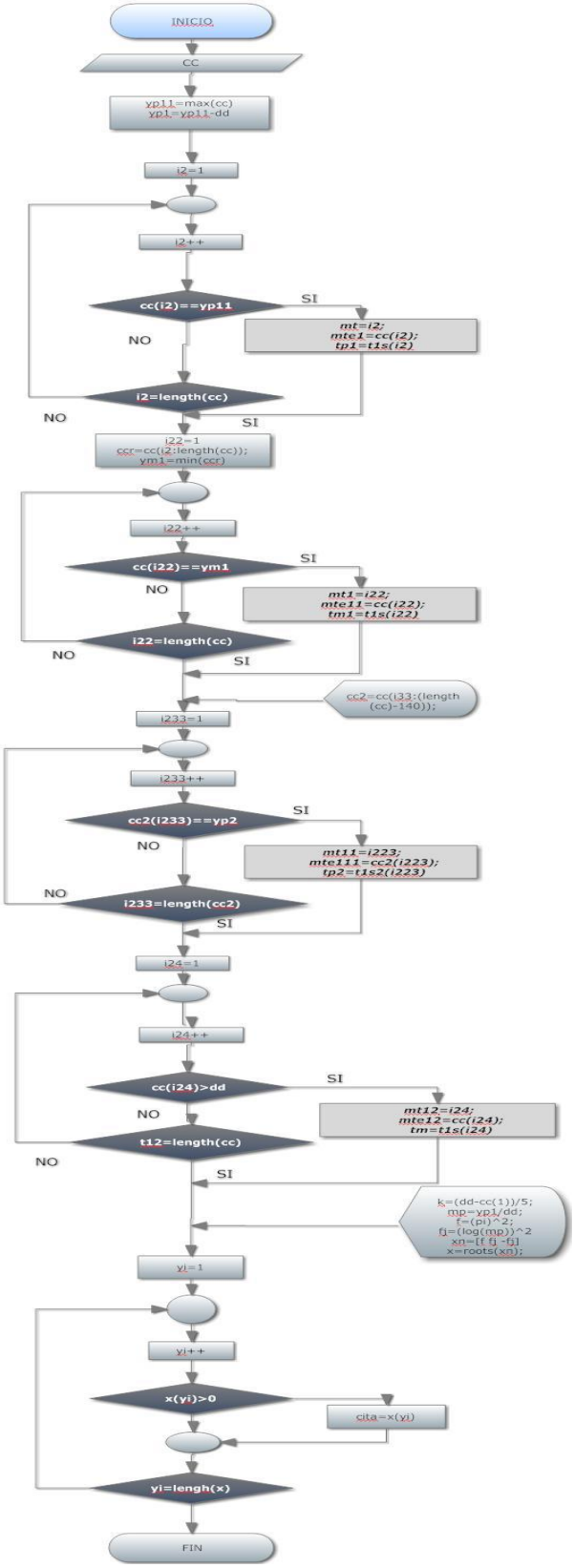
$$\frac{25.35}{s^2 + 2.363\ s + 5.469}$$



Algoritmo 4. Identificar 1



Algoritmo 5. Identificar 2



Algoritmo 6. Identificar 3

8.4 STEP MODELO IDENTIFICADO

Se aplica al modelo identificado un STEP, se obtiene su respuesta y se compara con la respuesta a un STEP del sistema real. Su código es representado en el algoritmo 7.

- **PRIMER ORDEN**

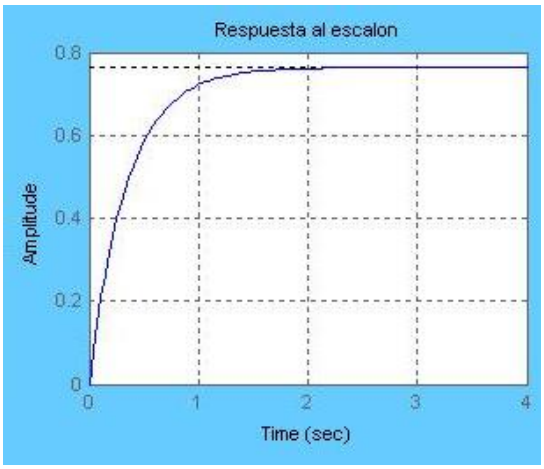


Figura 11. Respuesta al escalón (método Alfaro)

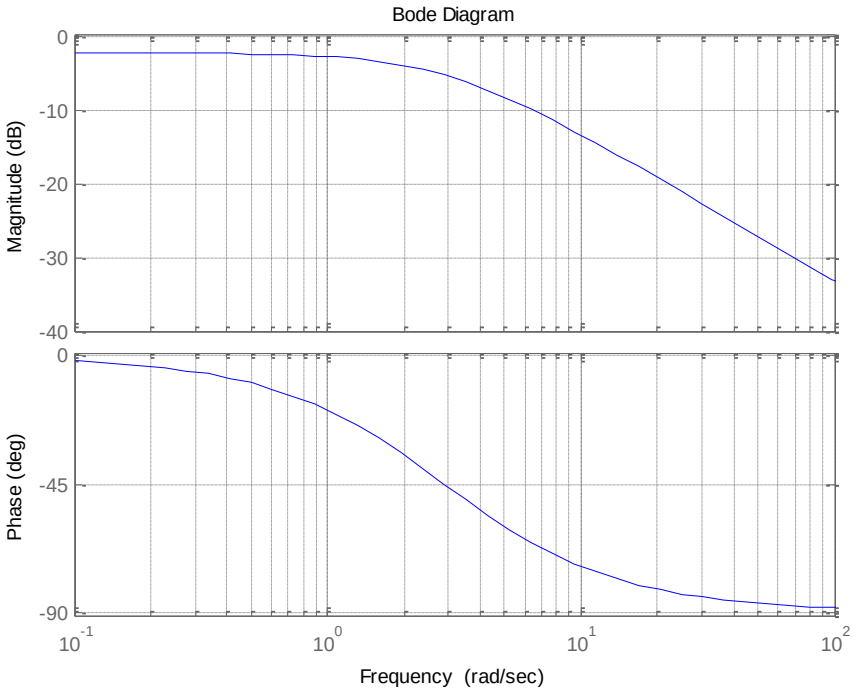


Figura 12. Diagrama de bode obtenido (primer orden)

- **SEGUNDO ORDEN**

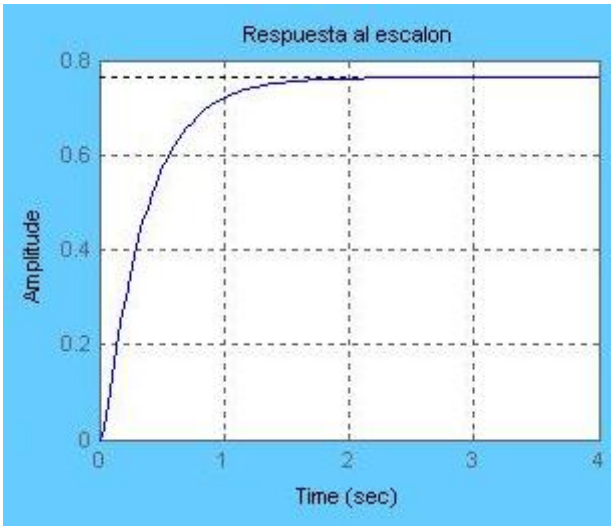


Figura 13. Respuesta al escalón (segundo orden, método Stark)

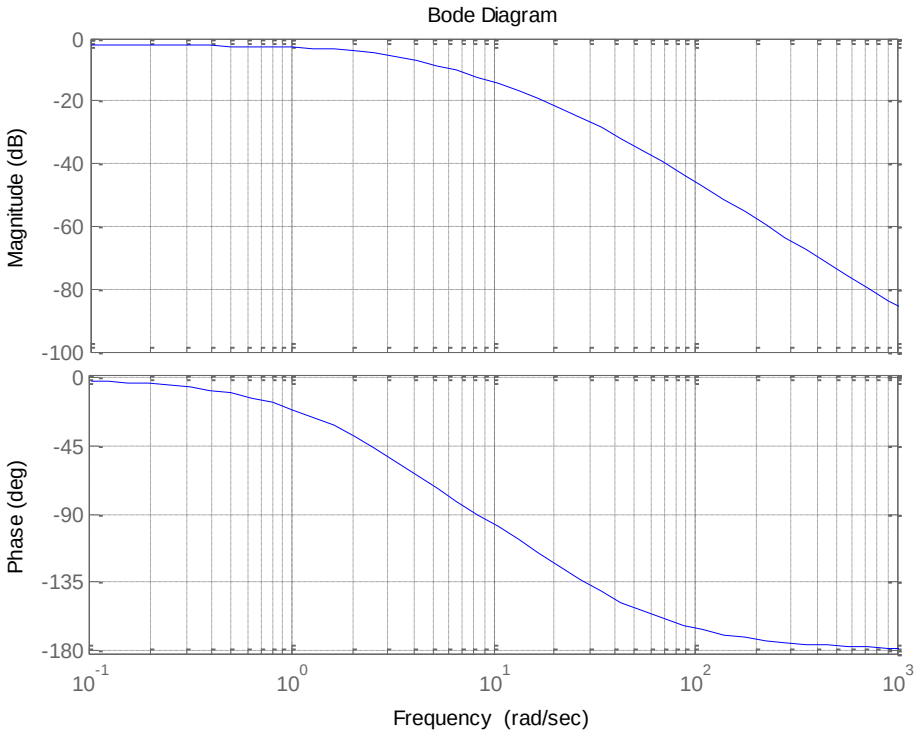


Figura 14. Diagrama de bode obtenido (segundo orden)

- **SUB-AMORTIGUADO**

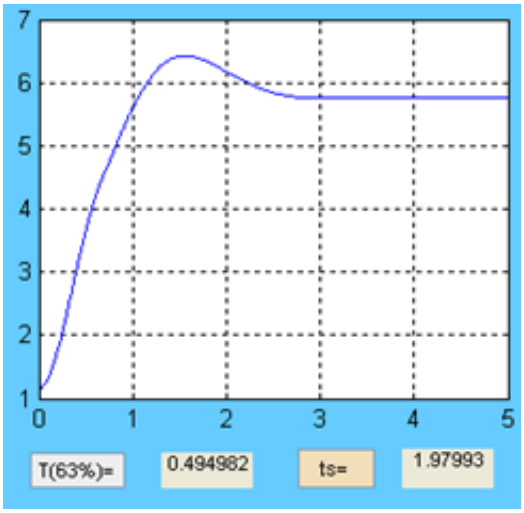


Figura 15. Respuesta al escalón (sub-amortiguado)

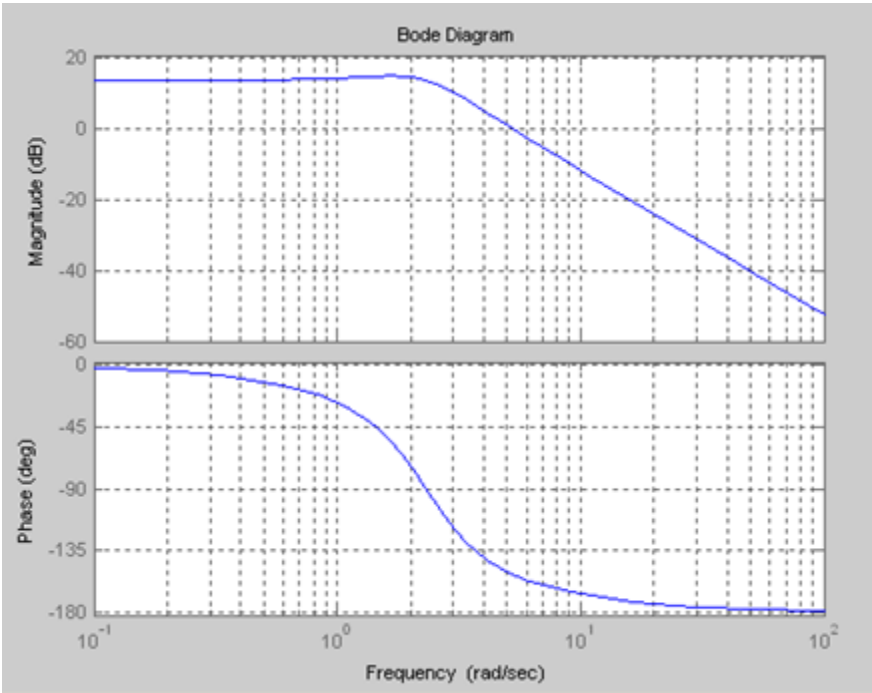
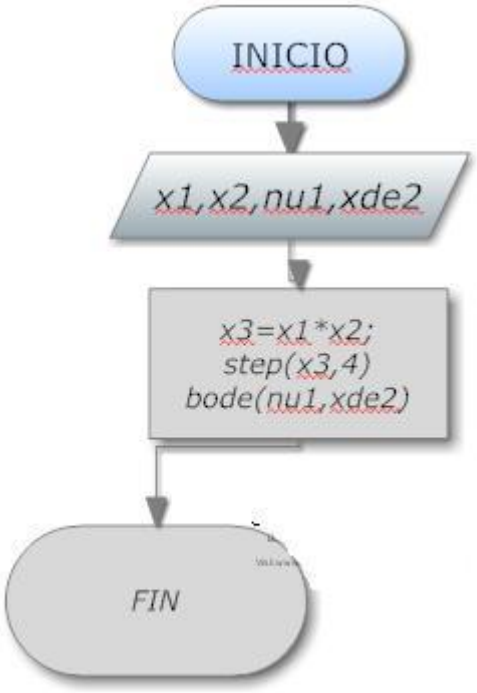


Figura 16. Diagrama de bode obtenido (sub-amortiguado)



Algoritmo 7. Step modelo identificado

8.5 CORRELACION LINEAL

Se mide el grado de intensidad de la relación entre la respuesta al STEP del modelo identificado y el sistema real. Se dibuja una vs la otra, y se encuentra su coeficiente de correlación. Como da cercano a 1, se demuestra que los datos son muy próximos a la del sistema real, por lo tanto posee una correlación lineal positiva (si sube el valor de una variable sube el de la otra).

La función que representa el coeficiente de correlación lineal en MATLAB es: $r = corrcoef(x, y)$, donde x y y son los vectores de datos.

El método de primer orden con el coeficiente de correlación más cercano a 1 es el método de Alfaro. Al graficar el error entre la respuesta de la planta y el método Alfaro tienen una relación lineal (ver figura 17), y su coeficiente de correlación es del 0.998333 (Vea figura 18). Su código es representado en el algoritmo 8.

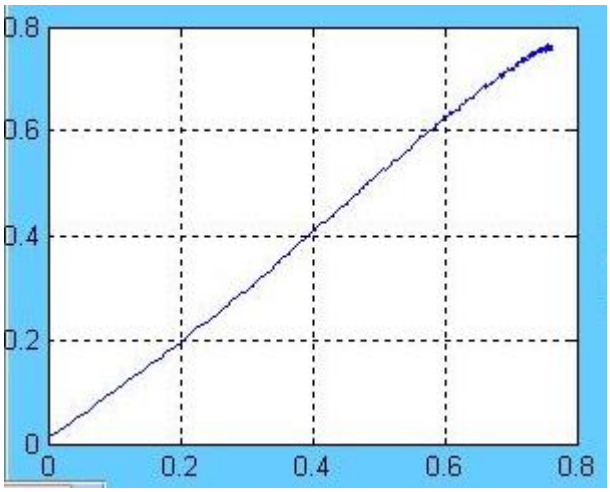
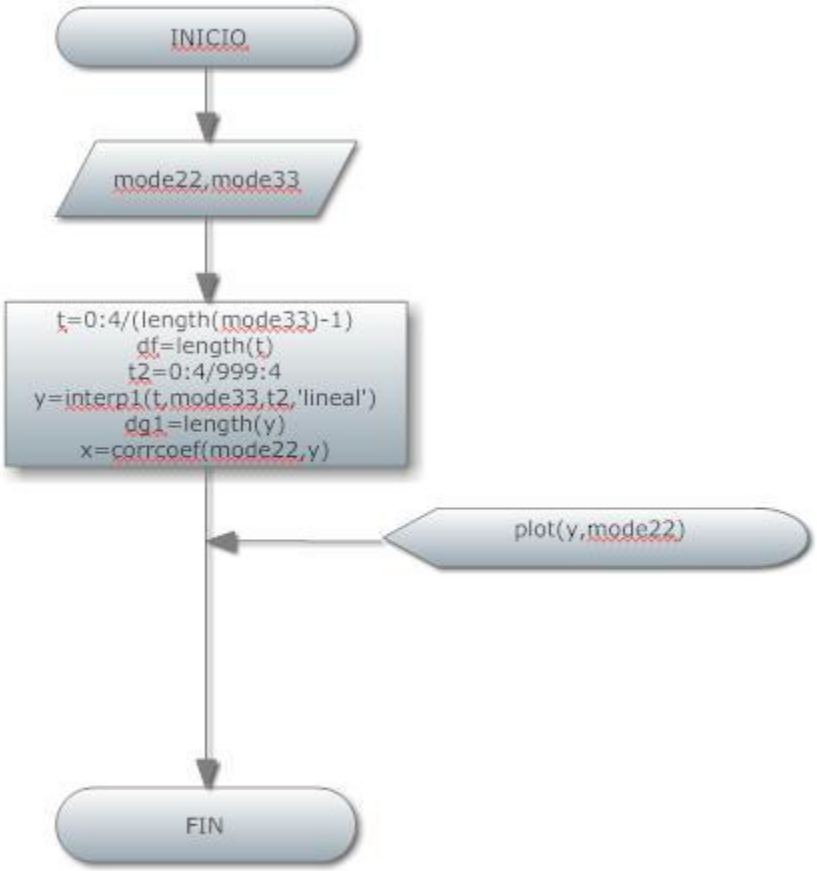


Figura 17. Correlacion lineal (Alfaro vs planta)

Figura 18. Coeficiente de correlacion

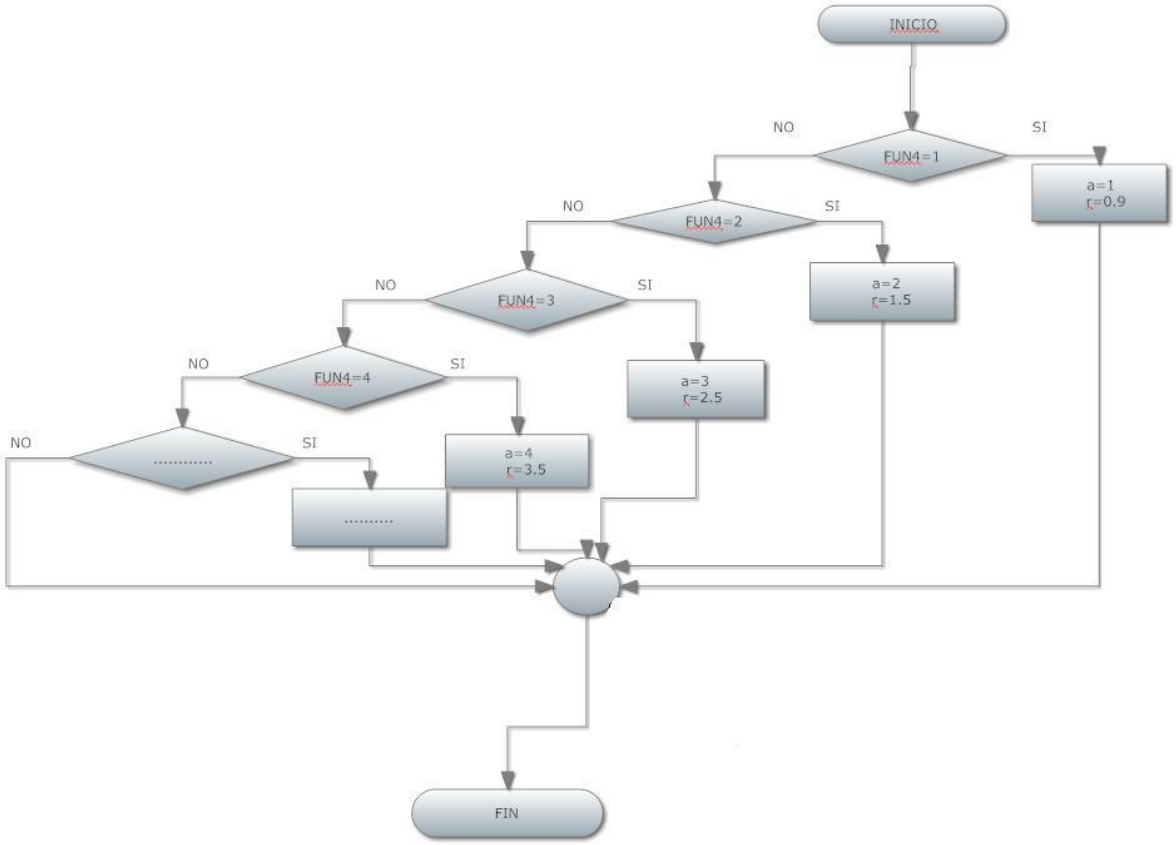


Algoritmo 8. Correlación

8.6 FRECUENCIAS UTILIZADAS

Se escoge las frecuencias en las cuales se va a trabajar, estas van desde 0.08hz hasta 75hz.

La frecuencia de la sinusoidal escogida para la demostración es de 0.08hz. Su código es representado en el algoritmo 9.



Algoritmo 9. Frecuencias utilizadas

8.7 LSIM, SIMULACION

Se simula el sistema identificado a entradas sinusoidales, para observar cómo se comporta frente a las señales sinusoidales recibidas de la planta, que en este caso es un motor dc. Ahí se corrobora que la señal de salida, a medida que aumenta la frecuencia, disminuye su amplitud y el desfase va aumentando (Vea Figuras 19 y 20). Su código es representado en el algoritmo 10.

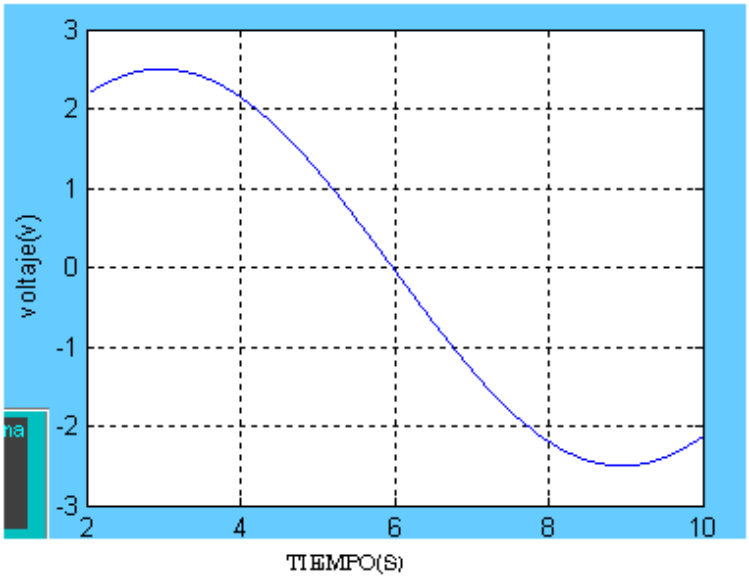


Figura 19. Simulacion de señal de entrada

Frec=0.08hz, amplitud=2.5v

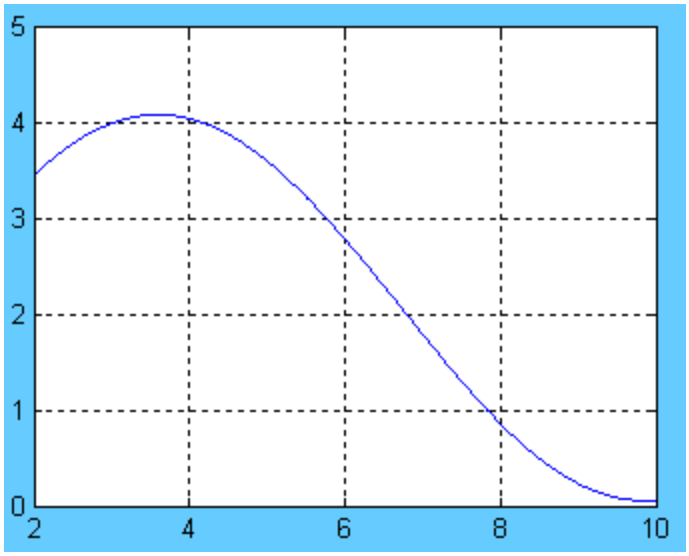
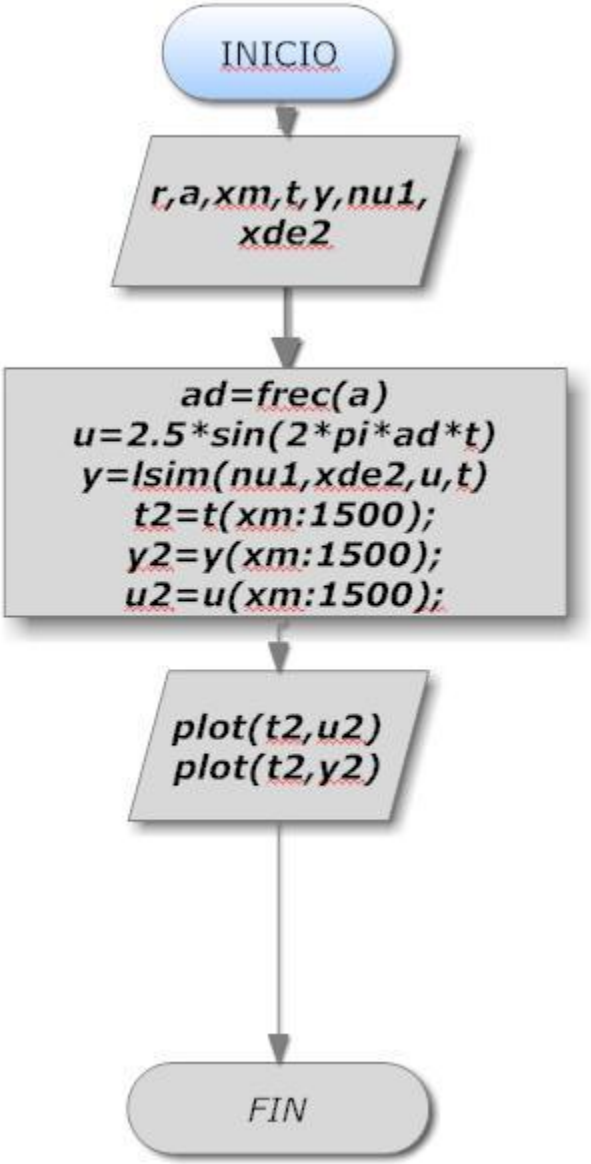


Figura 20. Simulacion de señal de salida

Frec=0.08hz, amplitud=2v



Algoritmo 10. Simulación de señal de entrada y salida

8.8 SALIDA, ENVIO DE SEÑAL AL MOTOR

Se envía la sinusoidal hacia el motor. Se obtiene el tiempo de cruce por 2.5 que se utiliza para encontrar el tiempo de retardo, se obtiene la frecuencia y su amplitud (Vea figura 21). Su código es representado en el algoritmo 11.

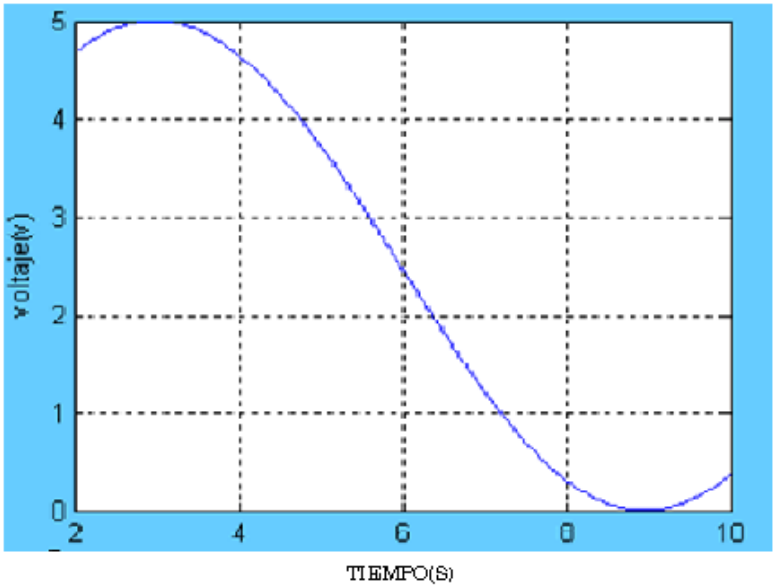
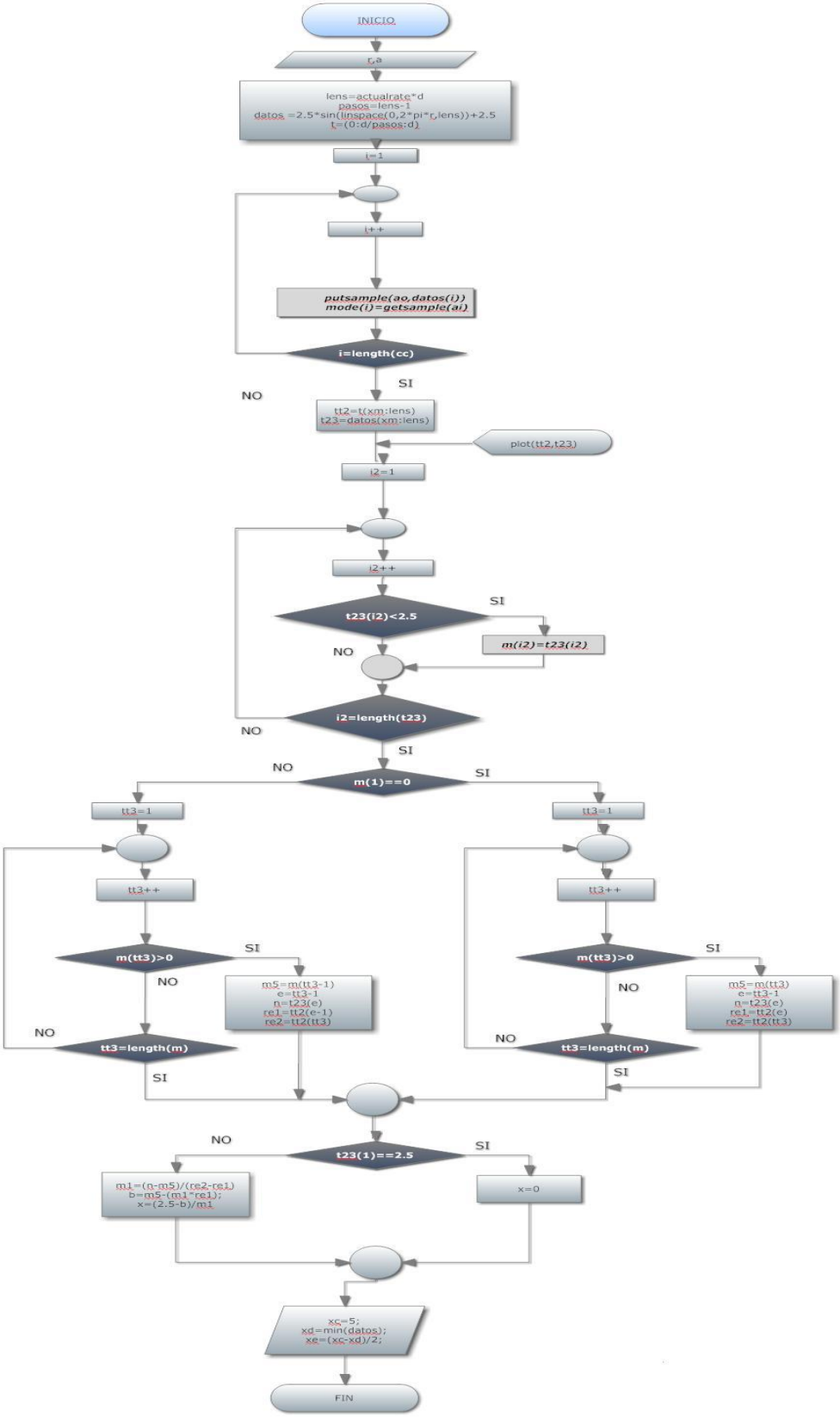


Figura 21. Señal enviada al motor

Frec=0.08hz, amplitud=2.5v



Algoritmo 11. Envió de señal al motor

8.9 ENTRADA, ADQUISICION DE LA SEÑAL GENERADA POR EL MOTOR

Se adquiere la señal que envía el motor (Vea Figura 22), esta es pasada obligatoriamente por un filtro de suavizado, ya que posee ruido blanco. El filtro que se utiliza es el SMOOT para poder suavizar la señal. Además de adquirir se encuentra su amplitud y el tiempo de cruce por 2v para obtener el retardo. Su código es representado en el algoritmo 12.

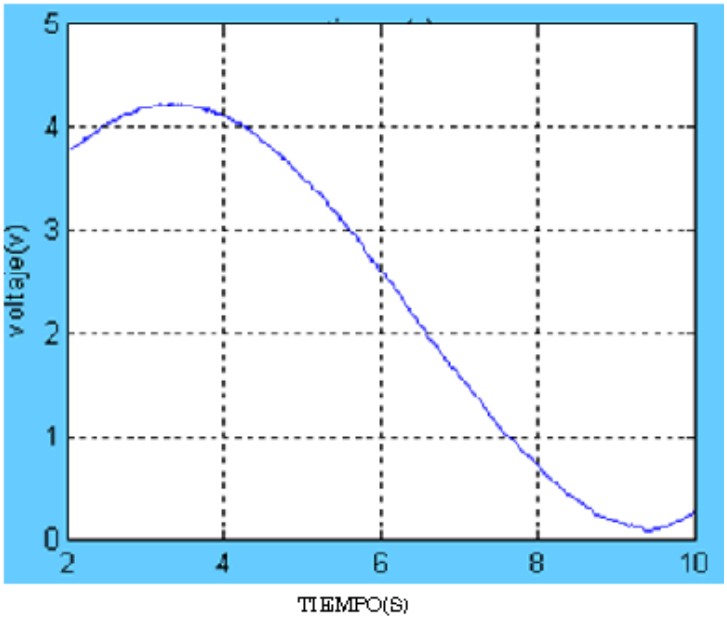
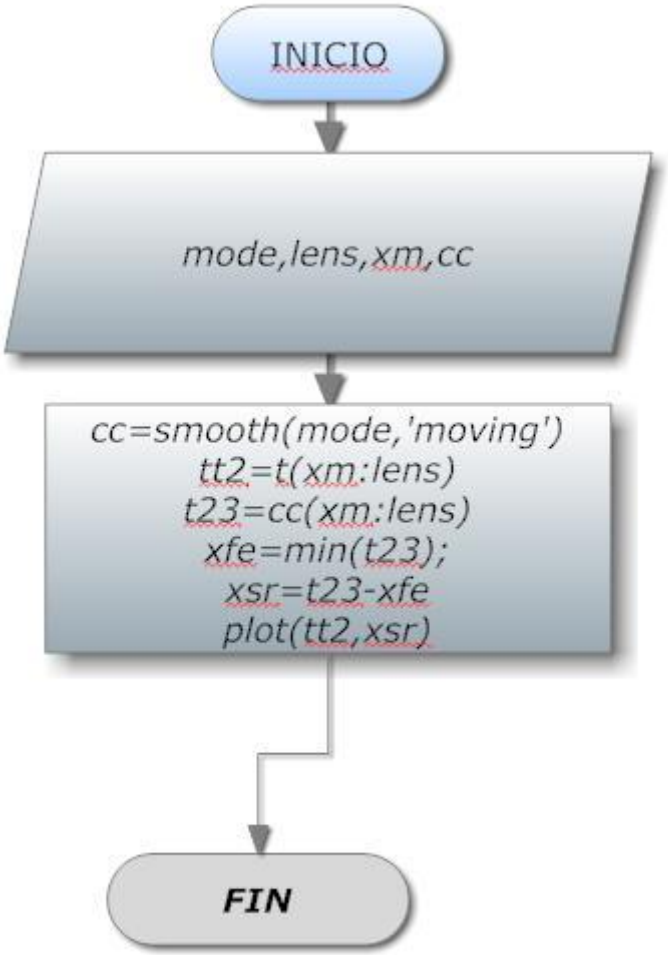


Figura 22. Señal adquirida del motor
Frec=0.08hz, amplitud=2v



Algoritmo 12. Adquisición de la señal generada por el motor

8.10 ANALISIS DEL ESPECTRO

Se encuentra el espectro de frecuencia de la señal adquirida del motor, con el proposito de que el usuario pueda visualizar que a medida que la frecuencia va aumentando, una señal de baja frecuencia se observa en el espectro. Debido a dicha frecuencia, es necesario aplicarse un filtro pasa altas para obtener una señal sin ruidos ni frecuencias que no pertenecen al espectro deseado.

8.10.1 FOURIER

Se encuentra el espectro de frecuencia de la señal, donde se observa las magnitudes de las frecuencias que la componen (Vea figura 23 y 24). Su código es representado en el algoritmo 13.

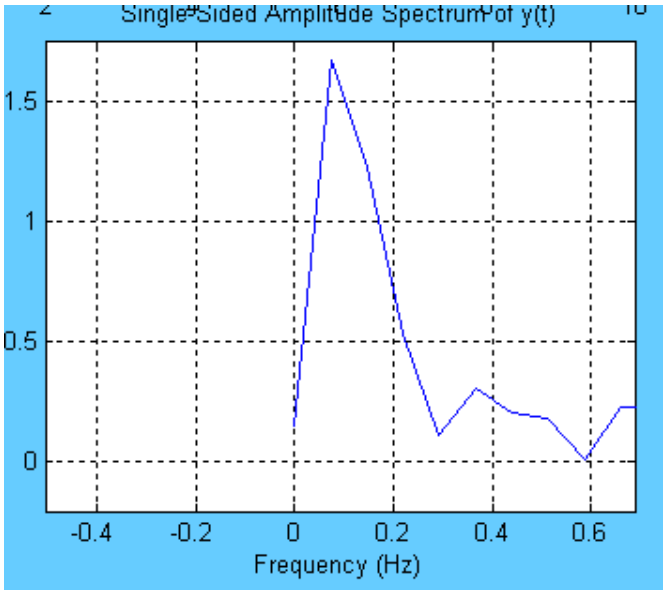


Figura 23. Espectro de frecuencia de la señal a 0.08 Hz

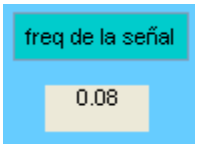
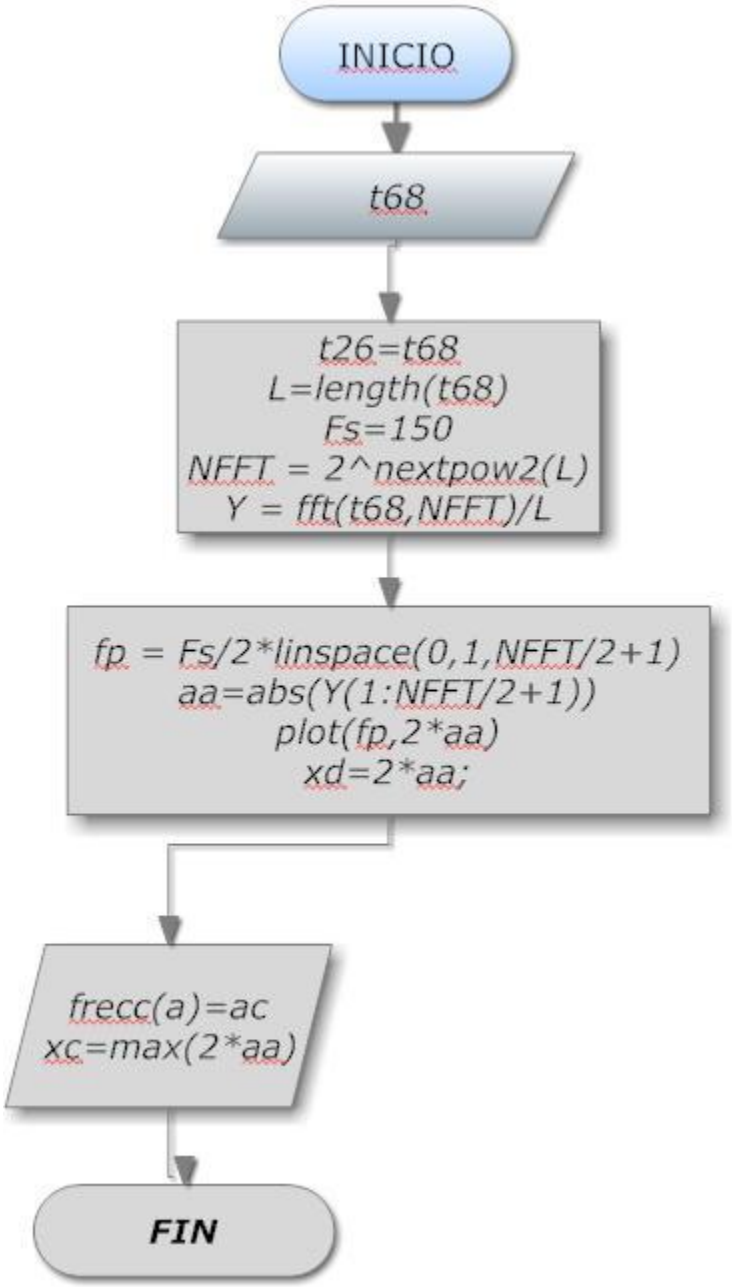


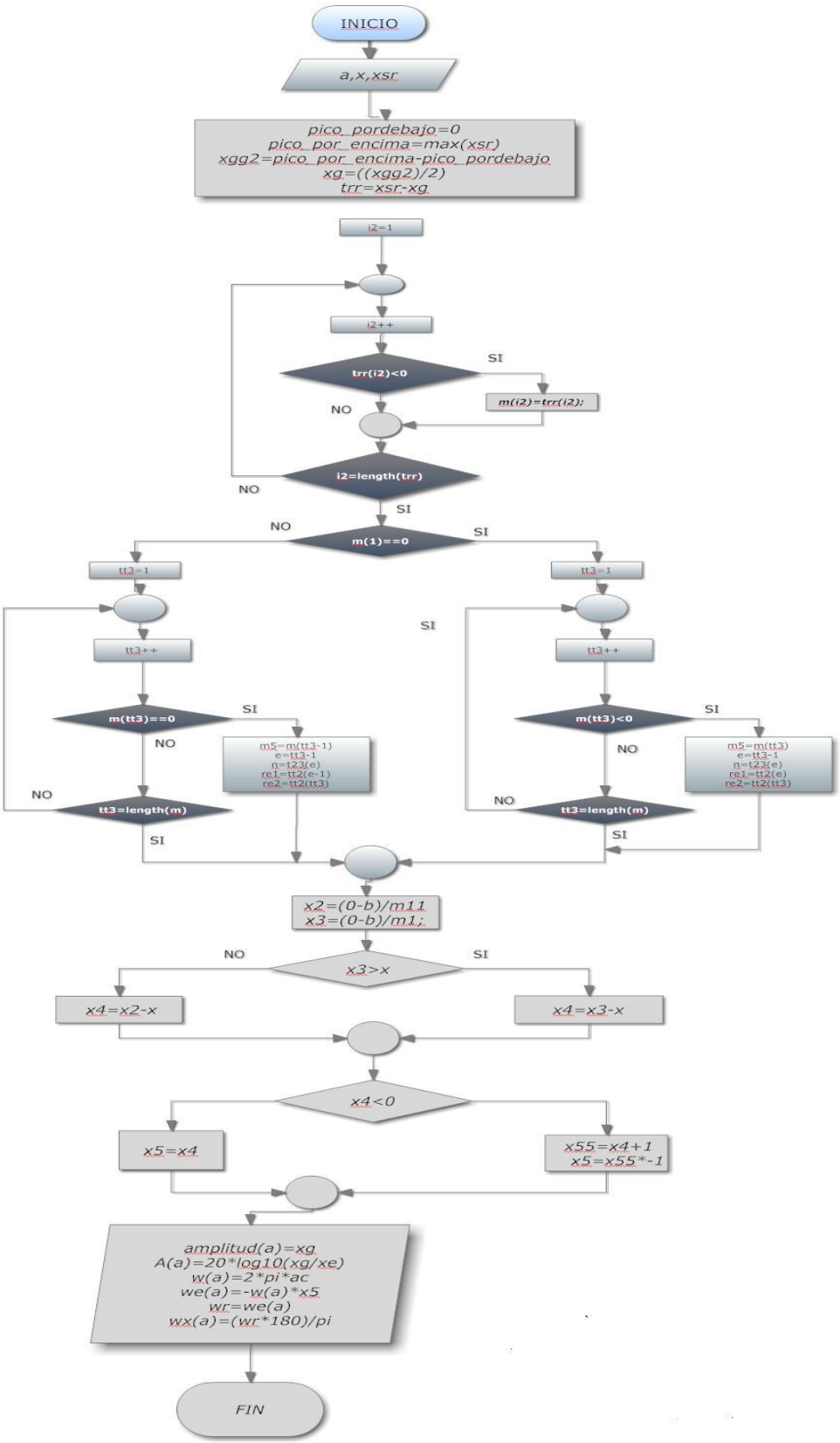
Figura 24. Frecuencia trabajada (0.08 Hz)



Algoritmo 13. **Fourier**

8.10.2 SIN FILTRO

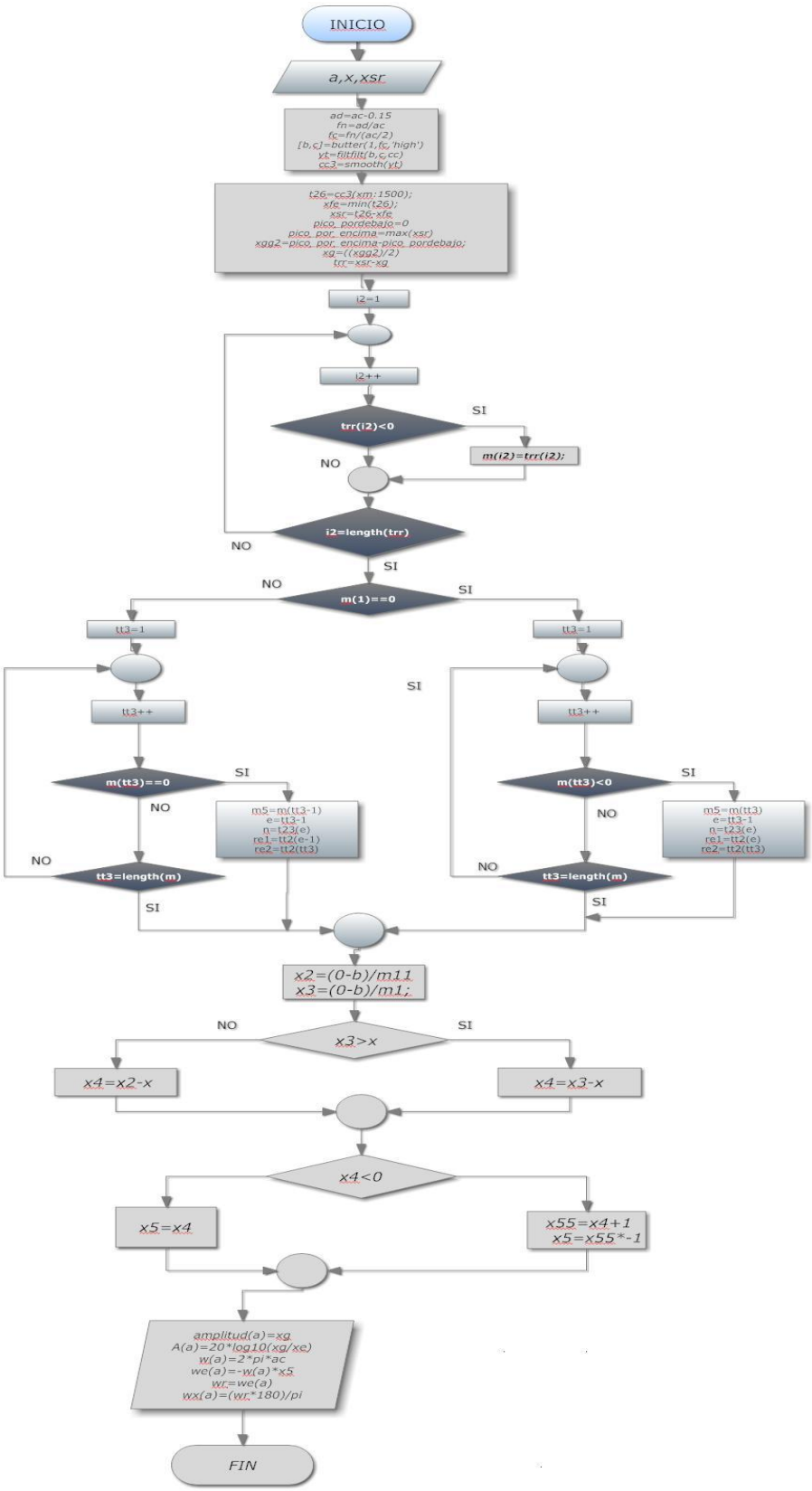
No a todas las frecuencias tienen la necesidad de aplicarle un filtro, por lo que el usuario puede decidir. El usuario visualizara las amplitudes de las señales enviadas y recibidas y el tiempo de retardo los cuales van a ir construyendo el diagrama de bode. Su código es representado en el algoritmo 14.



Algoritmo 14. Sin filtro

8.10.3 FILTRO PASA ALTAS

Se eliminan las bajas frecuencias que no correponden a la señal adquirida. Aquí igualmente se encuentra el tiempo de retardo, la amplitud de la señal filtrada, y la frecuencia que se elimina. El filtro que se usa para eliminar las bajas frecuencias es el filtro BUTTER en modo pasa alta. Su código es representado en el algoritmo 15.



Algoritmo 15. Filtro pasa altas

8.11 COMPARACION DE BODE

Grafica el diagrama de bode obtenido. Dibujando la magnitud y fase por separado. Se realiza la adquisición hasta una frecuencia de 0.5 hz o 3.1416 (rad/seg). Su código es representado en el algoritmo 16.

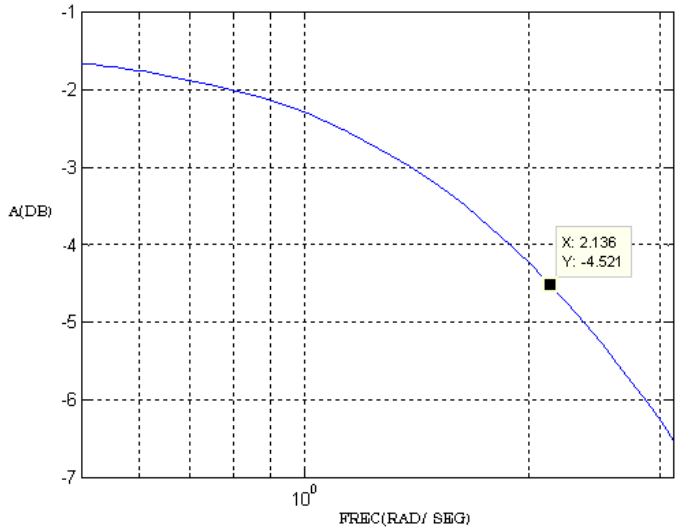


Figura 25. Diagrama de Bode respecto a la Magnitud del sistema obtenido

Se observa, que el polo estaría aproximadamente en 2.09, debido a que ya ha bajado el sistema 3db. Si se mira el bode del sistema identificado, se observa que el polo se encuentra casi en el mismo lugar, aproximadamente 2.08, y si se observa la respuesta al STEP del sistema real se ve que la constante del sistema tao está en 0.49, por lo que el polo estaría en 2.0408. Las figuras 25 y 26 muestran el diagrama de bode del sistema identificado

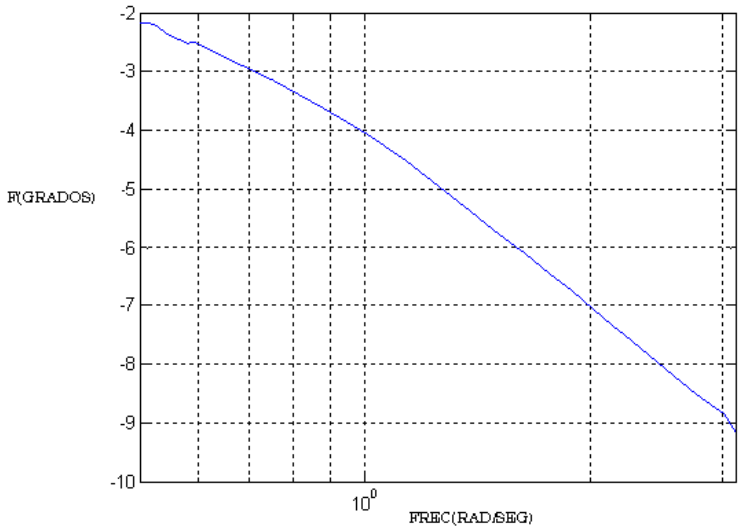
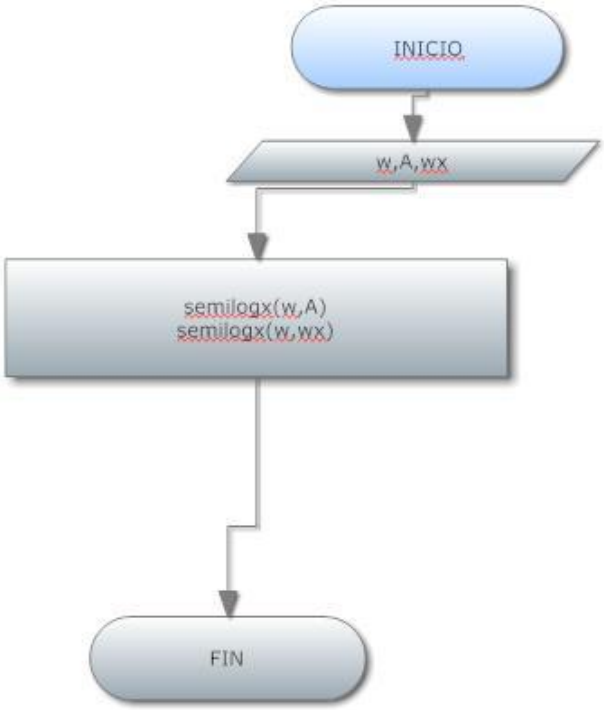


Figura 26. Diagrama de Bode respecto a la fase del sistema obtenido



Algoritmo 16. Diagrama de bode

8.12 FUNCION DE TRANSFERENCIA

Se obtiene la función de transferencia con la función Invfreqs de matlab. Se ingresa la magnitud y la fase compleja, la frecuencia (rads/seg), el orden del numerador y denominador. Su código es representado en el algoritmo 17.

- **PRIMER ORDEN**

$$\frac{2.186}{s + 2.872}$$

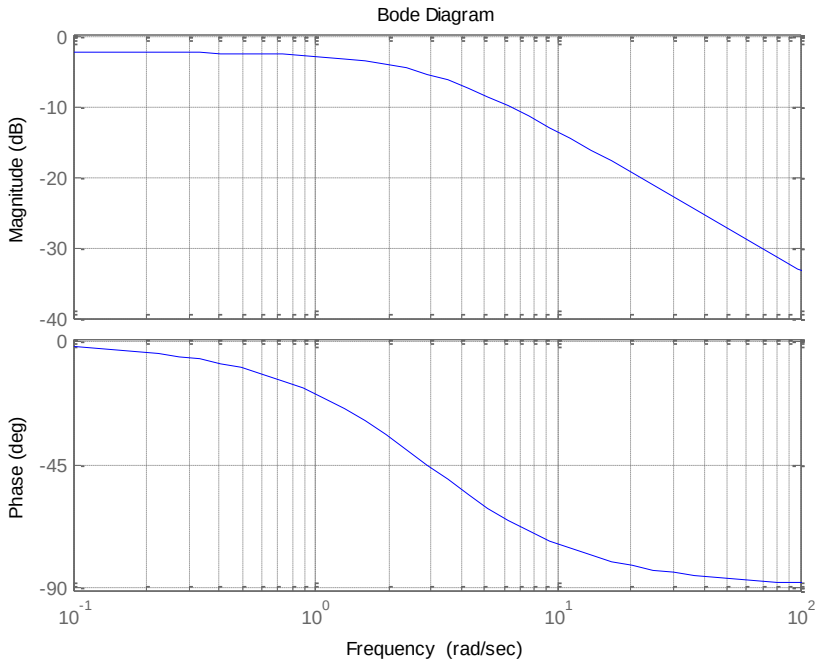


Figura 27. Diagrama de bode del sistema identificado (primer orden con la respuesta en frecuencia)

- **SEGUNDO ORDEN**

$$\frac{50.57}{s^2 + 25.18\,s + 66.45}$$

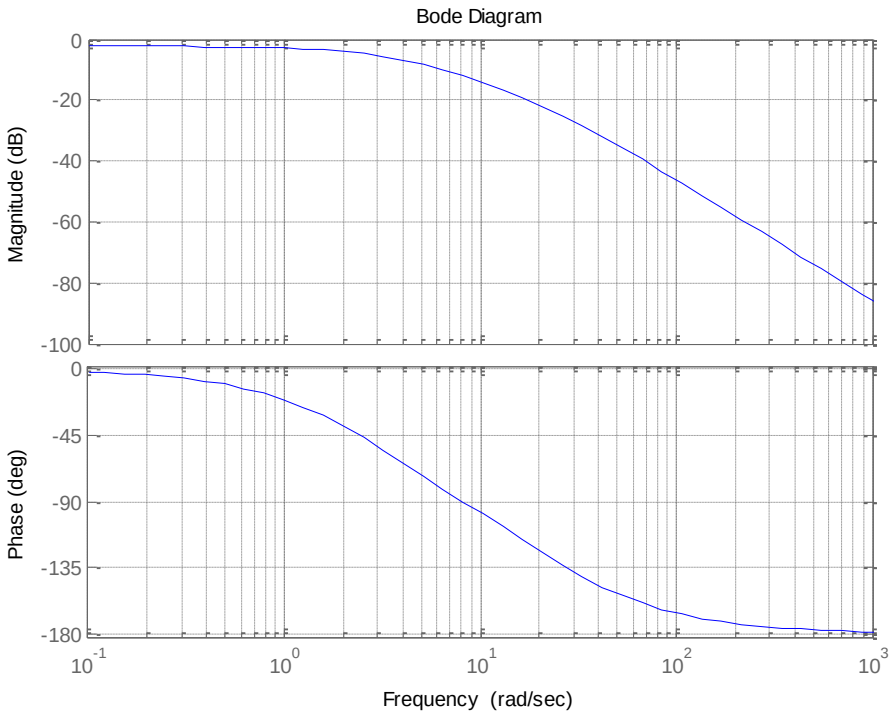
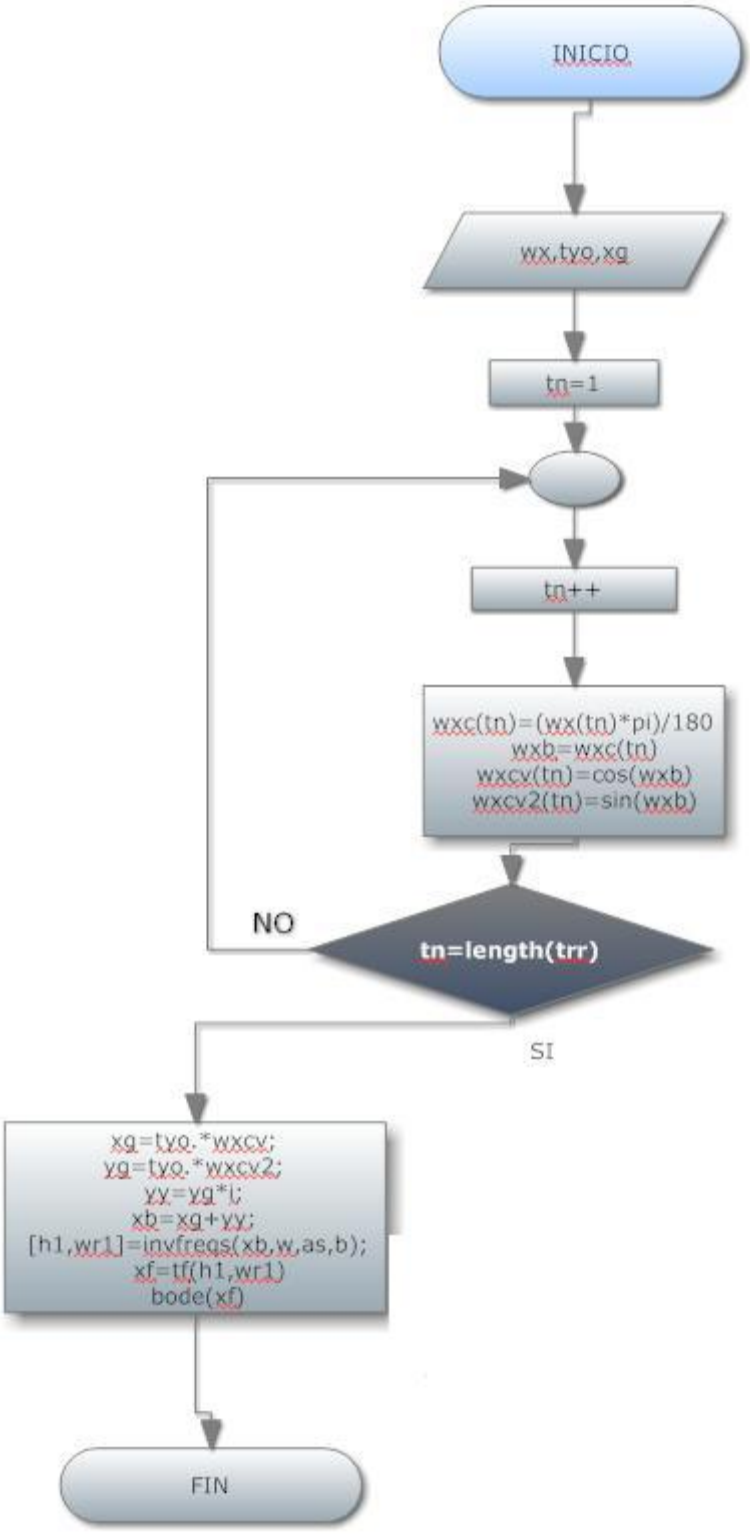


Figura 28. Diagrama de Bode del sistema identificado (segundo orden con la respuesta en frecuencia)



Algoritmo 17. Función de transferencia

8.13 STEP DEL MODELO OBTENIDO

Se encuentra la respuesta al escalón del modelo obtenido y la relación que tiene con respecto a la respuesta al escalón del modelo real. Su código es representado en el algoritmo 18.

- **PRIMER ORDEN**

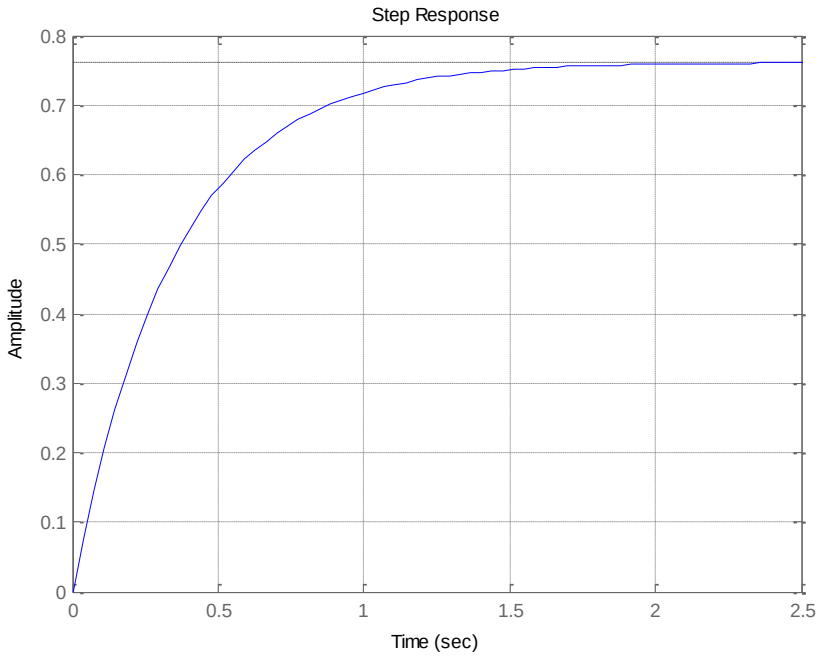


Figura 29. Step obtenido (primer orden)

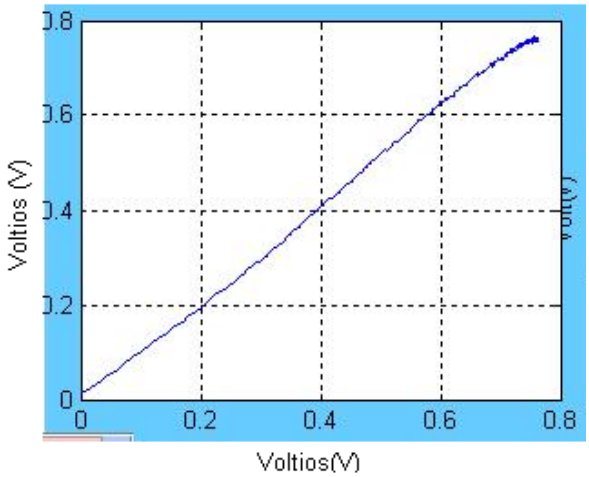


Figura 30. Correlacion lienal primer orden (obtenido vs real)

Figura 31. Coeficiente de correlacion (step obtenido vs real)

- **SEGUNDO ORDEN**

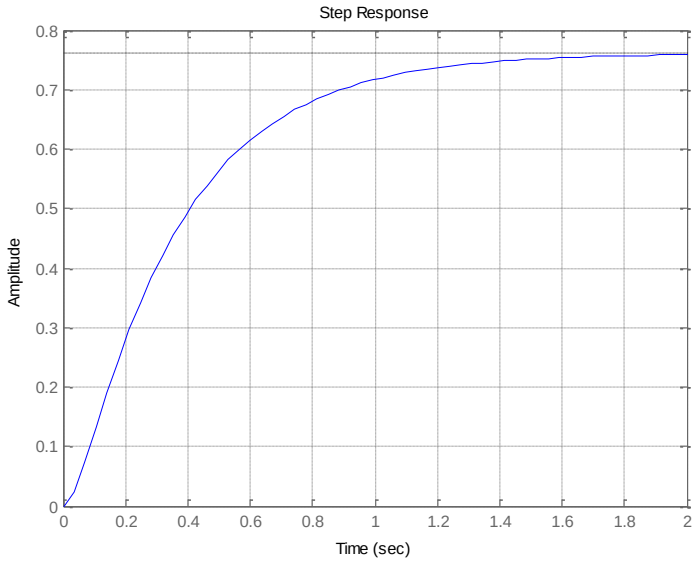


Figura 32. Step obtenido (segundo orden)

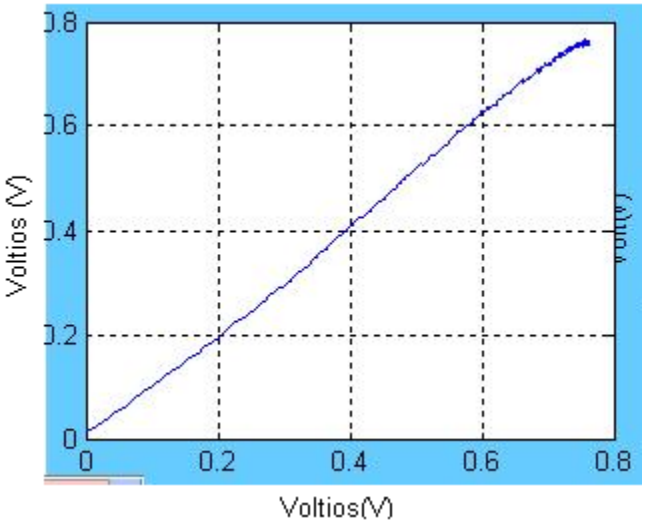


Figura 33. Correlación lineal segundo orden (obtenido vs real)

Figura 34. Coeficiente de correlacion (step obtenido vs real)

- **SUB-AMORTIGUADO**

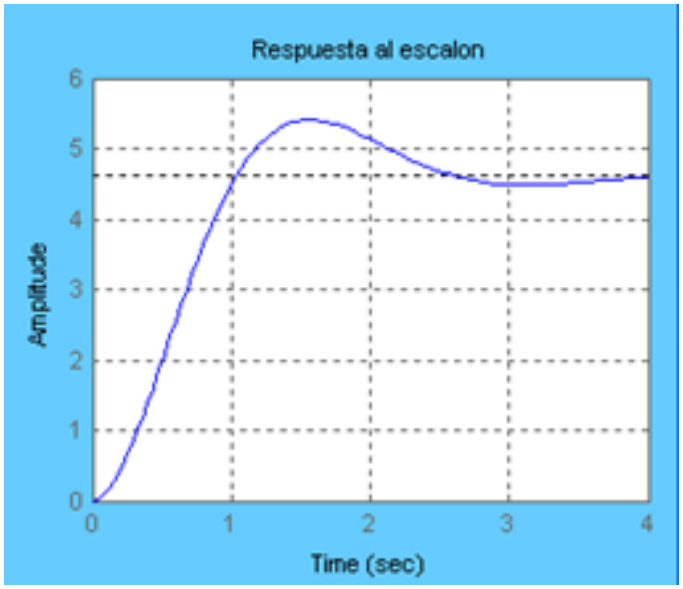


Figura 35. Step obtenido (sub-amortiguado)

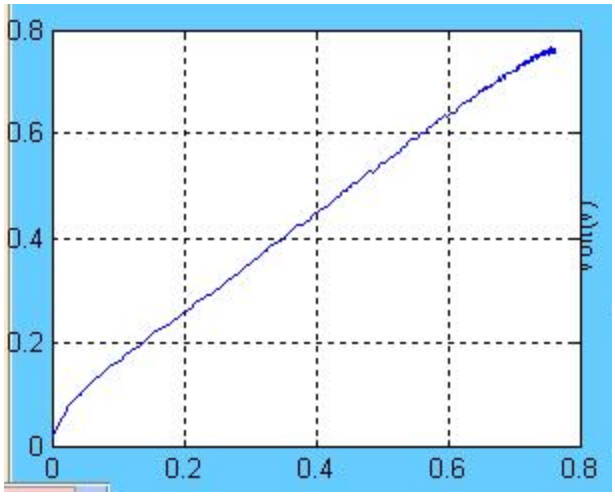
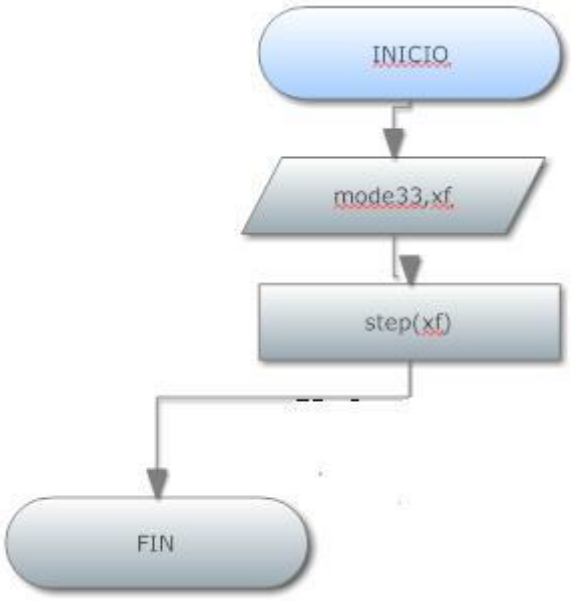


Figura 36. Correlación lineal Sub-amortiguado (obtenido vs real)

Figura 37. Coeficiente de correlacion (step obtenido vs real)



Algoritmo 18. Step del modelo obtenido

9. PUNTOS CRITICOS PARA PODER ADQUIRIR SEÑALES A TRAVEZ DE LA TARJETA USB 6008 Y MATLAB

- El sistema operativo debe ser xp.

- Remover la energía de las señales antes de conectarlas.

- Se debe conseguir un Matlab que contenga los adaptadores necesarios, en este caso se utilizo el R2009, de lo contrario no instala el adaptador de la tarjeta NI-DAQmx.

- Se debe crear los objetos de entrada y salida, ya sea análogo o digital, teniendo en cuenta que la tarjeta ya fue reconocida por el PC, y conociendo el ID de la tarjeta que puede ser DEV1, DEV2….

- Se debe configurar el modo de medición de la señal de entrada, ya sea nodo simple o nodo diferencial, se debe tener en cuenta que la elección depende de si la señal de entrada es referenciada a tierra o es una señal flotante. Como señal flotante tenemos termocuplas. Como señales referenciadas a tierra incluyen salidas de instrumentos no aislados y dispositivos que están conectados a una fuente de poder.

- Se debe tener cuidado con la tasa de muestro de los puertos de salida análogos, debido a la que la tarjeta 6008, tiene máximo una tasa de muestro de 150 muestras por segundo, en cambio en los puertos de entrada análogo la máxima tasa de muestreo es de 10000 muestras por segundo.

- Se debe tener en cuenta que la tarjeta 6008 tiene la desventaja de que posee un multiplexor que comparte el amplificador y el ADC con todos los canales, por lo que limita la velocidad de adquisición. Esto quiere decir que

si se va a usar un canal como entrada análoga se podrá adquirir a 10khz, pero si voy a usar 2 canales, la frecuencia de muestro se divide en los dos canales.

- La tasa de muestro de la señal de entrada análoga debe ser dos veces mayor al valor del componente de frecuencia máxima de la señal de entrada.

- Si se quiere adquirir y enviar señales análogas al mismo tiempo se debe utilizar el getsample y putsample, ya que se debe adquirir y recibir muestra a muestra, sino no se hace así, se envía pero no se adquiere la señal deseada.

- Si se va a adquirir señales digitales se debe utilizar el getdata, que extrae las muestras del motor de adquisición.

10. CONCLUSIONES

- Se realizo un libro disponible para la comunidad universitaria con la teoría de la respuesta en frecuencia, su obtención, su interpretación y resultados experimentales.

- Se le anexo al libro, un manual de MATLAB donde explica y muestra con ejemplos reales el envió y adquisición de datos análogos y digitales desde MATLAB.

- Se obtuvo el modelo del sistema a través de la respuesta en frecuencia, y además se anexo un manual del uso de la plataforma diseñada para llegar a obtener la respuesta en frecuencia.

- Se obtuvo el modelo del sistema a través de la respuesta en el tiempo. Para ello, se utilizaron diferentes métodos de identificación dependiendo del modelo seleccionado por el usuario.

- Se realizo una ponencia en la 3 semana nacional de ciencia, tecnología e innovación realizada en NEOMUNDO del 1 al 8 de octubre.

- La Toolbox de Adquisición de Datos de Matlab brinda soporte a las librerías y controladores de muchos dispositivos, generando así un fácil intercambio de información entre el software Matlab y el Hardware.

- Para realizar una sesión de adquisición de datos, independientemente de la aplicación que se tenga, se debe seguir una serie de pasos: se crea el objeto, se agregan las líneas o los canales al objeto, se configura el modo de medición (modo simple o modo diferencial), se envían o se extraen los datos y se borra el objeto si éste no se va a utilizar más.

-

BIBLIOGRAFIA

[1] VEGA URIBE, JESÚS ANTONIO. Matlab para Ingeniería. Universidad Pontificia Bolivariana, 2005.

[2] ROBERTS, MICHEL J. Señales y Sistemas: Análisis Mediante Métodos de Transformada y MATLAB. Editorial McGraw-Hill, 2005.

[3] SOLIMAN, SAMIR S. Y SRINATH, MANDYAM. Señales y Sistemas Continuos y Discretos. 2ª Edición. Editorial Prentice Hall, 1999

[4] THOMAS P. KRAUSS, LOREN SHURE. Signal Processing Toolbox, ejemplar 4

[5] EDWARD W. KAMEN. Introducción a Señales y Sistemas. 1ª Edición, México 1996.

[6] MAC E. VAN VALKENBURG. Design of Analog Filters, 2001

[7] R. Correa, J. Quiroz y R. Villamizar, "De la Sintonización de Controladores", Segunda Edición, Abril 2008, ISBN: 978-958-8187-82-2. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones, Universidad Industrial de Santander.

[8] www.gii.upv.es/filtrosdigitales.

[9] http://www.seh-lelha.org/tseries.htm

[10] http://www.dspguide.com/ch15.htm

[11] www.mathworks.com

[12] www.ni.com

[13] http://flops.wordpress.com/2007/03/15/interpolacion-lineal/

[14] www.ie.itcr.ac.cr/marin/lic/el3212/Diagrama_de_Bode.pdf

[15] http://isa.uniovi.es/docencia/adsii/practicasmodulos.pdf

ANEXOS 1

1. TARJETA NATIONAL INSTRUMENTS 6008

La tarjeta NI USB-6008 que se muestra en la Figura 38 proporciona la adquisición de datos a través del puerto USB. Para poder utilizarla con MatLab se debe tener instalado el controlador NI-DAQmx.

Este dispositivo está compuesto por 8 canales de entrada análoga (Ai) los cuales poseen una tasa de muestro de 10khz, es decir, 10.000 muestras por segundo. A su vez tiene 2 canales de salida análoga (AO), las cuales poseen una tasa de muestro máxima de 150hz, consigo tienen una resolución de 12 bits.

Esta es una tarjeta que tiene la desventaja de que posee un multiplexor que comparte el amplificador y el ADC con todos los canales, por lo que limita la velocidad de adquisición. Esto quiere decir que si se va a usar un canal como entrada análoga se podrá adquirir a 10khz, pero si voy a usar 2 canales, la frecuencia de muestro se divide en los dos canales.

Lo mismo sucede con la salida análoga, si uso los dos canales la frecuencia de muestro para cada canal será de 75hz. La resolución de la tarjeta varía dependiendo del modo de medición de entrada análoga.

En la medición por nodo simple, hay un cable de señal asociado a cada señal de entrada y cada señal de entrada está conectada a la misma tierra de la tarjeta, por este tipo de medición la tarjeta tiene una resolución de 11 bits. El rango de voltaje en nodo simple es de ±10v.

En la medición por nodo diferencial, hay dos cables de señal asociados con cada señal de entrada: uno para la señal de entrada y el otro para la señal de referencia (señal de retorno).

La medición es la diferencia de tensión entre los dos terminales. Este tipo de medición maneja una resolución de 12 bits. El rango de voltaje en nodo diferencial es máximo ±20v. Para la conversión A/D la tarjeta utiliza el convertidor por aproximaciones sucesivas.

Figura 38. Tarjeta de Nacional Instruments USB-6008

Fuente: Diseño de autor

1.1 SOFTWARE

El software que soporta USB-6008 para Windows 2000/XP es NI-DAQmx (adaptador que pertenece a la toolbox de adquisicion de datos de Matlab).

1.1.1 ENTRADA/SALIDA DIGITAL

La Tarjeta cuenta con dos puertos digitales, el puerto 0 de 8 canales y el puerto 1 de cuatro canales.

Tanto las entradas como las salidas digitales pueden ser tomadas como un bus de datos o como una línea individual, teniendo en cuenta que cuando se toma un bus de datos, las líneas contenidas por este deben ser configuradas como un puerto; es decir, todas deben tener la misma dirección de entrada o de salida.
La Tarjeta no permite que las líneas en un objeto si son de un mismo puerto sean configuradas individualmente.

- **SALIDA DIGITAL**

Para realizar una adquisición de datos como salida digital se debe llevar a cabo los siguientes pasos:

- **CREAR EL OBJETO:**

Los objetos son los elementos básicos de la Toolbox utilizados para acceder al dispositivo. Estos permiten el control del comportamiento y de la funcionalidad de la adquisición.

Los objetos digitales están asociados con los subsistemas de entrada/salida digital.

Se crean con la función **DIGITALIO**

Sintaxis:

$$DIO = digitalio('adaptador', ID)$$

DIO: Objeto digital

Adaptador: Nombre del adaptador. El adaptador para los dispositivos de adquisición de datos de la National Instruments se conoce como nidaq.

ID: Número asociado con el Hardware. Normalmente y según el número de dispositivo que se tengan instalados, para los de National Instruments el ID es una cadena de caracteres Dev1.

$$DIO = digitalio('nidaq', Dev1)$$

Al crear el objeto, automáticamente se establecen los valores de las propiedades básicas que son el tipo de objeto y el nombre.

- **AGREGAR LINEAS:**

Después de crear un objeto debe agregársele líneas. Las líneas son los elementos básicos con los cuales se adquiere o entrega datos. Si la salida está dada como línea individual se le agrega tan sólo una línea, pero si se requiere tomarla como bus de datos se debe agregarle dos o más líneas.

Cuando se agregan líneas a un objeto ya sea de salida o entrada digital, se deben cumplir las siguientes reglas:

- Las líneas deben ser del mismo hardware. No se puede agregar líneas de dispositivos diferentes, o de diferentes subsistemas del mismo dispositivo.

- Se puede agregar una línea sólo una vez a un objeto dado.

- Se puede agregar líneas de puertos diferentes, siempre y cuando los puertos sean del mismo dispositivo.

Sintaxis:

Para agregar una línea al objeto digital se usa la función ADDLINE de cuatro formas diferentes. Esta función requiere que se le especifique el nombre del objeto, el número (ID) de la línea del hardware y su dirección.

$$addline(Objeto, IDlineal, 'Dirección')$$

Objeto: Nombre del objeto digital al cual se le va a agregar la línea

Dirección: Sentido de la línea (Salida o Entrada)

ID Líneas: IDs de las líneas del hardware, son valores numéricos asignados por el fabricante para identificar las líneas de sus dispositivos, para los de National Instruments empiezan desde 0 hasta el número total de líneas que tienen los puertos digitales del hardware, aunque si se quiere referenciar la línea, se debe hacer usando los índices de MATLAB (empiezan desde 1).
Opcionalmente, se puede especificar el ID del puerto, nombres de líneas y un argumento de Salida, de las siguientes formas:

$$addline(Objeto, IDlineas, Puerto, ' Direccion')$$
$$addline(Objeto, IDlineas, 'Direccion,' nombres')$$
$$addline(Objeto, IDlineas, Puerto, 'Direccion,' nombres')$$

Puerto: ID del puerto, se especifica cuando se desea agregar una o más líneas de un mismo puerto o también cuando se requiere agregar líneas de diferentes puertos. La Tarjeta cuenta con dos puertos, en los cuales el ID del puerto uno de 8 líneas es 0 y el ID del puerto dos de 4 líneas es 1. Las líneas de estos puertos deben ser configuradas todas con la misma dirección.

Nombres: Nombre de las líneas, se especifica cuando se quiere referenciar con un nombre una o todas las líneas.

- **CONFIGURACIÓN DE LA SALIDA DIGITAL INDIVIDUAL**

Para configurar una salida digital como línea individual se debe tomar tan sólo un ID de cualquier línea de un puerto (ver anexo 1).

- **CONFIGURACIÓN DE LA SALIDA DIGITAL COMO BUS**

A diferencia de la salida individual, para configurar las salida digital como bus se le debe agregar al objeto dos o más líneas de cualquiera de los puertos de la Tarjeta teniendo en cuenta que todas las líneas deben tener la misma dirección ('out')(ver anexo 2 y anexo 3).

- **ENVIAR (ESCRIBIR) DATOS**

REGLAS PARA ESCRIBIR VALORES DIGITALES

- Se puede especificar los datos como un valor decimal o un vector binario. Un vector binario (o binvec) se construye con el bit menos significativo (LSB) en la primera columna y el más significativo (MSB) en la última columna. Por ejemplo, el decimal número 23 se escribe como el vector binario [1 1 1 0 1].

- Si un objeto contiene las líneas de un dispositivo de puerto configurable (caso de la Tarjeta) entonces todas las líneas serán escritas incluso si no están contenidas por el objeto.

- Un error es retornado si los datos son escritos en una línea de entrada.

- No se puede escribir un valor negativo, de lo contrario un error es retornado.

- Si un valor decimal es demasiado grande para ser escrito en la tarjeta un error es retornado.

Sintaxis:

Para escribir un dato se hace directamente con la función PUTVALUE sin necesidad de configurar alguna propiedad.

$$putvalue(Objeto, datos)$$

Objeto: Nombre del objeto digital creado y al cual se le agregó la línea como salida.

Datos: Números decimales o vectores binarios.

- **SALIDA DIGITAL INDIVIDUAL**

Para un objeto de salida digital individual, existe la posibilidad de escribir un solo valor binario o un vector de valores binarios.

$$putvalue(DIO, 0) \quad \text{ó} \quad putvalue(DIO, 1)$$

Si se quiere escribir un conjunto de datos es necesario crear un vector con valores binarios y luego escribir dato por dato dentro de un ciclo.
Con el objeto DIO conteniendo un solo canal de cualquier puerto los datos serían enviados así:

$$Datos = [1\ 0\ 0\ 0\ 1\ 1\ 1];$$
$$n = 0.5;$$
$$for\ i = 1{:}\ length(Datos)$$
$$putvalue(DIO, Datos(i))$$
$$pause(n)$$
$$end$$

Nota: Con la función pause se puede dar un tiempo de espera de n segundos entre cada valor del vector de datos para que estos no sean enviados rápidamente y puedan ser vistos (este paso es opcional).

- **SALIDA COMO BUS DE DATOS**

Cuando se realiza una adquisición de datos con una salida digital en bus, se puede enviar un número decimal, un Binvec o un vector de números decimales teniendo en cuenta que los números que se envíen deben ser de igual tamaño al número de las líneas agregadas.

Para enviar un solo número ya sea decimal o un Binvec se realiza directamente con la función PUTVALUE.
Por ejemplo, para enviar el decimal 23 o su respectivo Binvec con el objeto DIO con 4 canales del puerto 0, se hace de esta forma:

$$putvalue(DIO, 23) \quad \text{ó} \quad putvalue(DIO, [1\ 1\ 1\ 0\ 1\ 1]\ )$$

Para este ejemplo se debe tener en cuenta que el número de líneas es cuatro, es decir que el decimal o el vector binario deben ser de igual número de dígitos, esto quiere decir que cualquier número desde el 0 hasta el 15 podría ser enviado.

Con el mismo objeto DIO de 4 líneas, si ahora lo que se quiere enviar es un vector de números decimales:

$$datos = [10\ 4\ 6\ 7\ 8\ 9];$$
$$n = 0.5;$$
$$for\ i = 1{:}\ length(datos)$$
$$putvalue(DIO, datos(i))$$
$$a = getvalue(DIO)$$
$$pause(n)$$
$$end$$

- **BORRAR EL OBJETO**

Este paso se realiza cuando no se necesita por mucho tiempo el objeto o se quiere crear otro objeto en el mismo dispositivo.

Sintaxis:

Para remover el objeto tanto del área de trabajo como del dispositivo, primero se usa la función DELETE seguida del comando CLEAR.

delete(OBJ)
clear

OBJ: Nombre del objeto que va a ser removido.

La función DELETE elimina el objeto y las líneas contenidas por éste del motor de adquisición del dispositivo, debe ser usada únicamente al final de una sesión de adquisición.
El comando CLEAR remueve todos datos del área de trabajo de MATLAB

1.1.2 ENTRADA DIGITAL

Para realizar la lectura de datos digitales con entradas individuales o como bus por medio de la Tarjeta, se debe seguir los siguientes pasos.

- **CREAR EL OBJETO**

Los objetos de entrada digitales, ya sean como entrada individual o como entrada en bus se crean de igual forma como se crean los objetos de salida digital. Esto es con la función DIGITALIO y con los mismos argumentos de entradas.

$$DIO = digitalio('adaptador', ID)$$

$$DIO = digitalio('NIDAQ', Dev1)$$

- **AGREGAR LINEAS**

Para agregarle líneas a un objeto de entrada digital se usa la función ADDLINE, con dirección de entrada (in), la sintaxis que se emplea es la misma para agregar líneas a un objeto de salida digital teniendo en cuenta que ahora se pretende leer datos, es decir, los datos se adquieren desde el exterior hacia el entorno de MATLAB.

Sintaxis:

$$addline(Objeto, IDlineas, 'Direccion')$$
$$addline(Objeto, IDlineas, Puerto, 'Direccion')$$
$$addline(Objeto, IDlineas, 'Direccion,' nombres')$$
$$addline(Objeto, IDlineas, Puerto, 'Direccion,' nombres')$$

- **CONFIGURACIÓN DE ENTRADA DIGITAL INDIVIDUAL**

Al ser una entrada digital individual, al objeto DIO se le agrega tan solo una línea (ver anexo 4).

- **CONFIGURACIÓN DE ENTRADA DIGITAL EN BUS**

Al ser una entrada digital en bus, al objeto DIO se le agrega varias líneas (ver anexo 5 y anexo 6).

- **EXTRAER (LEER) DATOS**

Se puede leer datos digitales de una o más líneas con la función GETVALUE. Los valores leídos son tomados como valores como un vector binario (Binvec), opcionalmente se puede hacer la conversión a decimal con la función BINVEC2DEC.

REGLAS PARA LEER VALORS DIGITALES

Si un objeto de entrada/salida digital contiene líneas de un dispositivo de puerto configurable, entonces todas las líneas son leídas, incluso si están contenidas por el objeto, sin embargo, sólo los valores de las líneas contenidas por el objeto son retornados.

Para los dispositivos de National Instruments usando la Interfaz tradicional NI-DAQ, las líneas configuradas para entrada por defecto retornan un valor 1.

GETVALUE siempre retorna un Binvec.

Sintaxis:

$$Valor = getvalue(Objeto)$$

Valor: Nombre de la variable en la cual se guarda el valor digital extraído.

Objeto: Nombre del objeto de entrada digital creado con anterioridad.
Este objeto debe contener líneas con dirección de entrada. En caso de no ser así no es posible realizar la lectura.

- **ENTRADA DIGITAL INDIVIDUAL**

--> Para extraer el valor de una línea individual contenida por un objeto digital DIO, se realiza con la función GETVALUE, proporcionando el nombre del objeto.

$$Dato = getvalue(DIO);$$

- **ENTRADA DIGITAL EN BUS**

Los datos de un objeto que contiene varias líneas de entrada, se extraen de igual manera como se hace para los que tiene una línea de entrada individual. Aunque es posible extraer los valores de una o de varias líneas en particular, siempre y cuando estén contenidas por el mismo objeto (ver anexo 7).

- **BORRAR EL OBJETO**

Cuando no se necesite más el objeto, como último paso en una adquisición, se procede a borrar el objeto del dispositivo y del área de trabajo de MATLAB con las funciones DELETE y CLEAR, usadas respectivamente.

delete (DIO)
clear OBJ

1.1.3 SALIDA ANÁLOGA

Un subsistema de salida análoga convierte los datos digitales almacenados en el computador en datos análogos.

La tarjeta posee un terminal de dos salidas análogas con un rango máximo de tensión entre 0 y 5 Voltios.
Para acceder a los objetos de salida análoga, se debe seguir los siguientes pasos:

- **CREAR EL OBJETO DE SALIDA ANÁLOGA**

Para crear el objeto se hace de la siguiente manera:

$$ao = analogoutput('adaptador','ID')$$

Adaptador: Nombre del adaptador. El adaptador para los dispositivos de adquisición de datos de la National Instruments se conoce como nidaq.

ID: Número asociado con el Hardware. Normalmente y según el número de dispositivo que se tengan instalados, para los de National Instruments el ID es una cadena de caracteres Dev1.

- **AGREGAR CANALES AL OBJETO**

Después de crear el objeto, se debe agregarle canales.
La tarjeta tiene dos terminales para salida análoga, por lo tanto un bus de salida puede contener sólo este número de canales.
Para agregarle canales a un objeto de salida análoga hay que tener en cuenta la siguiente regla:

- Los canales deben ser del mismo hardware. No se puede agregar canales de diferentes dispositivos o de diferentes subsistemas del mismo dispositivo.

Sintaxis:

Los canales son agregados al objeto con la función ADDCHANNEL. Esta función requiere el objeto y por lo menos un ID de canal como argumentos de entrada. Opcionalmente se podría especificar los nombres descriptivos de canal, los índices de MATLAB y un argumento de salida o variable para acceder posteriormente a las propiedades de los canales.

$$Chan = addchannel(Objeto, IDcanales)$$
$$Chan = addchannel(Objeto, IDcanales, indice)$$
$$Chan = addchannel(Objeto, IDcanales, 'nombre')$$
$$Chan = addchannel(Objeto, IDcanales, indice, 'nombre')$$

Chan: Argumento de salida, nombre que se le da a la variable con la cual se accede a las propiedades de los canales agregados.

Objeto: Nombre del objeto de salida análoga al cual se le agrega canales.

ID canales: IDs de los canales del hardware, son valores numéricos asignados por el fabricante para identificar los canales de sus dispositivos, para los de National Instruments empiezan desde 0 hasta el número total de canales de salida análoga que posee la tarjeta. En este caso los IDcanales son 0 y 1 respectivamente.

Índices: Índices de MATLAB, se usan para referenciar los canales con la nomenclatura de MATLAB, la cual comienza en 1 y van incrementando de uno en uno hasta el número de canales contenidos por el objeto, siguiendo el orden en que son agregados los canales.

Nombre: Nombre de los canales, se especifica cuando se quiere referenciar con un nombre una o todas los canales del objeto.

- **CONFIGURACIÓN SALIDA ANÁLOGA INDIVIDUAL**

Para agregar el canal 1 a un objeto de salida análoga (AO) creado con anterioridad, con el nombre Entrada, se realiza con estos argumentos de entrada para la función ADDCHANNEL.

$$Objeto = AO$$
$$IDcanal = 1$$
$$Nombre = Entrada$$
$$Chan = addchannel(AO, 1, 'Entrada');$$

El índice de MATLAB para este canal es establecido automáticamente en 1.

- **CONFIGURACIÓN SALIDA ANÁLOGA EN BUS**

La tarjeta cuenta con dos terminales de salida análoga, por lo tanto, para tomar una salida en bus, ésta debe contener los dos canales (ver anexo 8).

- **ENVIAR (ESCRIBIR) LAS MUESTRAS**

La tarjeta está limitada a escribir directamente una muestra (dato) análoga por canal ya que esta no cuenta con una fuente de reloj, quien se encarga de la sincronización de la tarjeta con el computador, por lo tanto, no es necesario configurar propiedades de muestreo. Para escribir una señal completa pero muestreada, se debe hacer mediante un ciclo repetitivo o un temporizador.

Sintaxis:

Con la función PUTSAMPLE se escribe una muestra por canal, requiriendo como argumentos de entrada el objeto y la muestra que se va a escribir.

$$putsample(Objeto, dato)$$

PUTSAMPLE no almacena los datos en el motor de adquisición de datos; esta función se puede usar en cualquier momento, siempre y cuando se haya agregado canales al objeto de salida análoga.

- **SALIDA ANÁLOGA INDIVIDUAL**

--> Para enviar un valor de tensión análogo a un objeto de salida análoga individual, por ejemplo 3.8V, se toma los siguientes argumentos.

$$Objeto = AO$$
$$dato = 3.8$$

$$putsample(AO, 3.8)$$

--> Si los datos son una cadena de valores análogos, se escriben mediante un ciclo repetitivo.

$$Objeto = AO$$
$$Datos = 0{:}\,0.2{:}\,5$$
$$datos = 0{:}\,0.2{:}\,5$$
$$for\ i = 1{:}\,length(datos)$$
$$putsample(AO, datos(i))$$
$$pause(0.1)$$
$$end$$

- **SALIDA ANÁLOGA EN BUS**

--> Para escribir en un objeto de salida análoga en bus, dos valores de tensión diferentes, cada uno por un canal, se usa la función PUTSAMPLE de la siguiente forma:

$$Objeto = AO$$
$$Datos = [4\ \ 2]$$
$$putsample(AO, [4\ 2])$$

Los datos están dentro de un vector fila, el cual consiste en una muestra por canal contenida por el objeto. Es decir, la muestra de la primera columna (4) es escrito en el canal 0 y la de la segunda columna (2) en el canal 1.

--> Si los datos que se van a escribir son un arreglo de valores, esto debe hacerse dentro de un ciclo.

$$Objeto = AO$$
$$Datos = [4\ \ 2]$$
$$datos = 0{:}\ 0.2{:}\ 5$$
$$for\ i = 1{:}\ length(datos)$$
$$putsample(AO, [datos(i)\ datos(i)])$$
$$pause(0.1)$$
$$end$$

Cuando no se necesite más el objeto, como último paso en una adquisición, se procede a borrar el objeto del dispositivo y del área de trabajo de MATLAB con las funciones DELETE y CLEAR, usadas respectivamente.

- **BORRAR EL OBJETO**

Cuando no se necesite más el objeto, como último paso en una adquisición, se procede a borrar el objeto del dispositivo y del área de trabajo de MATLAB con las funciones DELETE y CLEAR, usadas respectivamente.

$$delete\ (OBJ)$$
$$clear\ OBJ$$

1.1.4 ENTRADA ANÁLOGA

Los subsistemas de entrada análoga, convierten una señal análoga en una señal digital que puede ser leída por el computador.
La Toolbox de Adquisición de Datos permite el acceso a las entradas análogas de los dispositivos a través de los objetos de entrada análoga.

La tarjeta cuenta con 8 terminales de entrada análoga con un rango de tensión de -10V a 10V para mediciones con entradas de nodo simple y de -20V a 20V para mediciones con entradas diferenciales.

Para realizar la adquisición de datos con entradas análogas se debe seguir los siguientes pasos:

- **CREAR EL OBJETO DE ENTRADA ANÁLOGA**

Para crear el objeto se hace de la siguiente manera:

$$ao = analoginput('adaptador','ID')$$

Adaptador: Nombre del adaptador. El adaptador para los dispositivos de adquisición de datos de la National Instruments se conoce como nidaq.

ID: Número asociado con el Hardware. Normalmente y según el número de dispositivo que se tengan instalados, para los de National Instruments el ID es una cadena de caracteres Dev1.

- **AGREGAR CANALES**

Los canales de un objeto de entrada análoga se agregan de la misma forma como se agregan los canales a los objetos de salida análoga.
Para esto se usa la función ADDCHANEL, especificando como argumentos de entrada para ésta función los IDs de los canales y el nombre del objeto al cual se le va a agregar los canales y opcionalmente se puede especificar los índices de MATLAB y algún nombre descriptivo para los canales.

- **CONFIGURAR PROPIEDADES**

Antes de leer los datos es necesario configurar algunas propiedades del objeto y de los canales contenidos por el objeto, aunque dependiendo de la configuración del dispositivo y de los requerimientos de la aplicación, es posible utilizar los valores por defecto de las propiedades y omitir este pasó.

Las propiedades de la Toolbox de Adquisición de Datos están divididas en dos tipos:

Propiedades comunes: Aplican a todos los canales o líneas contenidas por el objeto.

Propiedades de Canales o Líneas: Son configuradas para canales individuales.

La relación entre un objeto de entrada análoga, los canales contenidos por él y sus propiedades se muestran a continuación:
Para muchas aplicaciones comunes, hay un pequeño grupo de propiedades relacionadas con la configuración básica que se debe usar comúnmente. Estas propiedades de configuración básica controlan la tasa de muestreo, definen el tipo de entrada, el tipo de Trigger (disparador) y las muestras a adquirir por disparo.

Los valores de las propiedades se asignan usando la función SET o la notación punto.
Pueden ser configuradas en cualquier momento, sin embargo, algunas propiedades son configurables sólo cuando el objeto no está corriendo.
La función SET es usada también para retornar todas las propiedades configurables de un objeto y de sus canales, retornando todos los valores posibles para las propiedades.
Con la función GET se puede tomar el valor actual de una o más propiedades de un objeto o de los canales contenidos por el objeto.

- **TIPOS DE ENTRADA**

Una de las propiedades comunes es el tipo de entrada que se quiere tomar para la medición. Los valores de esta propiedad están dados según lo permitido por un hardware.
Por ejemplo para los dispositivos de National Instruments los posibles valores son los siguientes, aunque la tarjeta permite sólo realizar mediciones con entradas análogas en modo diferencial y de nodo simple.

Para saber el valor actual de inputType se usa la función GET de la siguiente forma:

$$get(AI, 'InputType')$$

Por defecto el valor de la propiedad del tipo de entrada es diferencial (Differential), pero se puede establecer para nodo simple si así lo requiere la aplicación.

$$set(AI, 'InputType', 'SingleEnded')$$

El valor de InputType determina el número de canales del hardware que pueden agregarse a un objeto. Si se toma el valor por defecto (Differential), el máximo número de canales que puede contener el objeto es 4, mientras que al establecer el valor de InputType para entradas de simple nodo (SimpleEnded) los 8 canales de entrada análoga que tiene la tarjeta pueden ser agregados al objeto.
Se debe tener en cuenta que la tensión de entrada en los canales para mediciones diferenciales se puede establecer en un rango máximo de -20V a 20V y para las de nodo simple en un rango máximo de -10V a 10V. El valor máximo de tensión de la señal no puede pasar estos rangos, si así fuera, la tarjeta satura los valores que se pasan hasta el valor máximo de tensión establecido

Esta propiedad es una propiedad de canal, por lo tanto se debe especificar el nombre del argumento de salida que se toma al agregar los canales.

Para entradas de Nodo simple

$$set(chan, 'InputRange', [-10\ 10])$$

Para entradas Diferenciales

$$set(chan, 'InputRange', [-20\ 20])$$

El rango de tensión puede ser también establecido independiente para cada canal sin pasar el rango máximo. Por ejemplo, para un objeto de entrada diferencial análoga en bus, con dos canales:

$$set(AI.channel(1), 'InputRange', [-20\ 20]\ )$$

$$set(AI.channel(2), 'InputRange', [-5\ 5]\ )$$

1.2 HARDWARE

En el siguiente diagrama de bloques (Vea Figura 39) se encuentran los principales componentes de las NI-USB 6008.

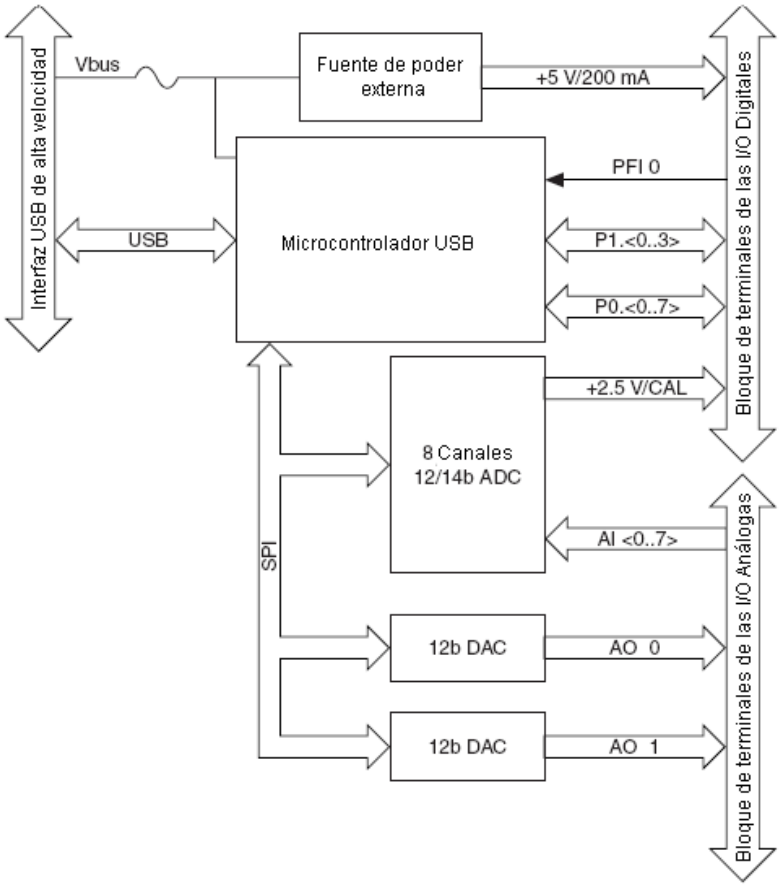


Figura 39. Hardware de la tarjeta

1.2.1 TERMINALES ANALOGAS Y DIGITALES

En la tarjeta National Instruments USB-6008 se encuentran dos diferentes bloques de termianles; un bloque de terminales para señales análogas y otro bloque para señales digitales.

Las Figuras 40 y 41 muestran la distribución de los terminales en la tarjeta.

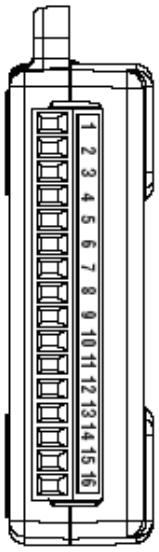
Module	Terminal	Signal, Single-Ended Mode	Signal, Differential Mode
	1	GND	GND
	2	AI 0	AI 0+
	3	AI 4	AI 0–
	4	GND	GND
	5	AI 1	AI 1+
	6	AI 5	AI 1–
	7	GND	GND
	8	AI 2	AI 2+
	9	AI 6	AI 2–
	10	GND	GND
	11	AI 3	AI 3+
	12	AI 7	AI 3–
	13	GND	GND
	14	AO 0	AO 0
	15	AO 1	AO 1
	16	GND	GND

Figura 40. Terminales análogos

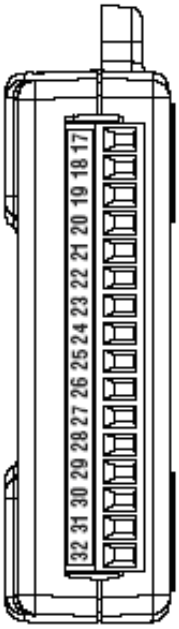
Module	Terminal	Signal
	17	P0.0
	18	P0.1
	19	P0.2
	20	P0.3
	21	P0.4
	22	P0.5
	23	P0 6
	24	P0.7
	25	P1.0
	26	P1.1
	27	P1.2
	28	P1.3
	29	PFI 0
	30	+2.5 V
	31	+5 V
	32	GND

Figura 41. Terminales digitales

- **DESCRIPCION DE TERMINALES**

AI<0….7> (entradas análogas): para medidas en modo nodo simple sólo son las entradas del canal análogo en voltaje. En modo diferencial AI0 a AI4 son entradas positivas y negativas para las entradas análogas diferenciales del canal 0.

AO0 (salida con referencia en GND, salida análoga 0): suministra un voltaje de salida para el canal 0.

AO1 (salida con referencia en GND, salida análoga 1): suministra un voltaje de salida para el canal1.

P1<0….3> y P0<0….7> (referencia en tierra): Puertos de entrada o salidas digitales. Cada puerto se puede configurar como entrada o salida individualmente.
+2.5V (referencia en GND): referencia externa de 2.5Voltios para pruebas en realimentación.

+5V (referencia en tierra): fuente de 5v, a 200mA (salida).

GND: punto de referencia para las medidas de entrada análoga de simple nodo, voltajes de salida, señales digitales en los conectores de I/O, fuente de +5v, y en los +2.5vdc de referencia.

1.2.2 ENTRADAS ANALOGAS

Las señales de entrada análoga son señales físicas que pueden ser generadas por un sensor o un generador de funciones, son continuas en el tiempo y en amplitud (dentro de límites predeterminados). La Figura 42 muestra un circuito de entradas análogas.

Circuito:

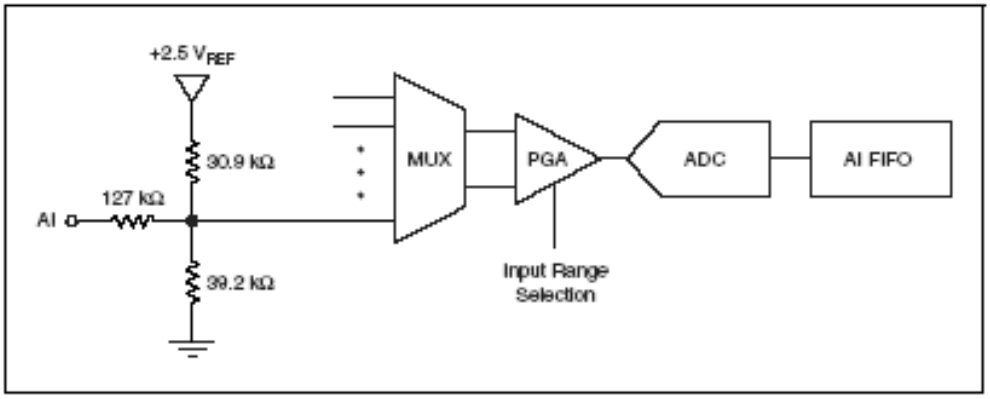


Figura 42. Circuito de entradas análogas

- **Mux***:*

La USB 6008 tiene un convertidor análogo a digital. El multiplexor enruta un canal de AI a tiempo con la PGA.

- **PGA**:

Amplificador de ganancias programables. Provee ganancia de entradas1, 2, 4, 5, 8, 10, 16, o 20 para la configuracion en medidas diferenciales y ganancia unitaria cuando se configuran para entrada sencilla. La ganancia del PGA es automaticamente calculada según el rango de voltaje seleccionado en la aplicación.

- **A/D convertidor*:***

El convertidor digitaliza la señal de entrada análoga convirtiendo el voltaje análogo en un código digital.

- **AI FIFO:**

La tarjeta USB puede realizar conversiones A/D simples y multiples de codigo externo.

- **MODOS DE ENTRADAS ANALOGAS**

Existen dos formas de medir las entradas análogas, esto depende si la señal de entrada es referenciada a tierra o es una señal flotante. Como entrada de nodo simple (monopolar) y como entrada diferencial.

- **MODO DIFERENCIAL**

Para lograr medir las entradas análogas en modo diferencial se debe conectar el pin positivo de la señal al terminar AI+ y el pin negativo de la señal al terminal AI- (Vea Figura 43).

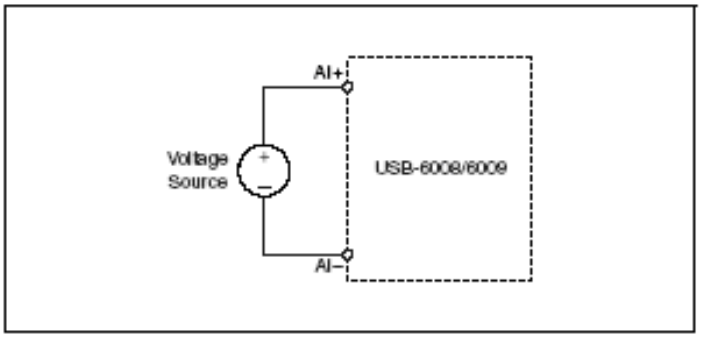


Figura 43. Esquema de modo diferencial

Al tener las entradas análogas en el modo de entradas diferenciales se puede llegar a medir señales de ±20v en el rango de más o menos ±20v (Vea Figura 44). Sin embargo, se debe tener en cuenta que el máximo voltaje en cualquier pin es de ±10v con respecto a tierra (Vea Figura 45). Por ejemplo si AI1 es +10v y AI5 es de -10v, se encuentra que la medida tomada por el dispositivo es de +20v.

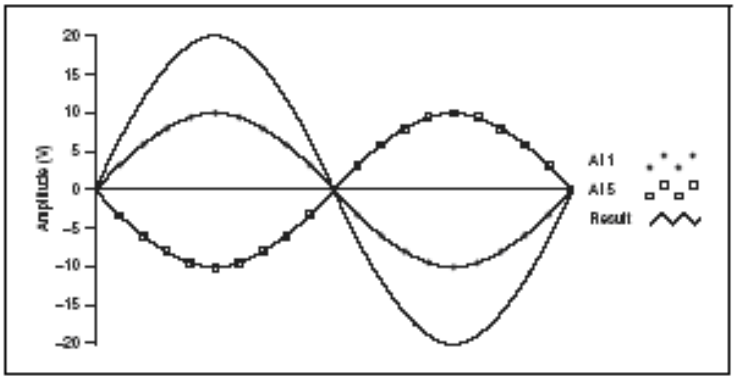


Figura 44. Señal en modo diferencial 20v

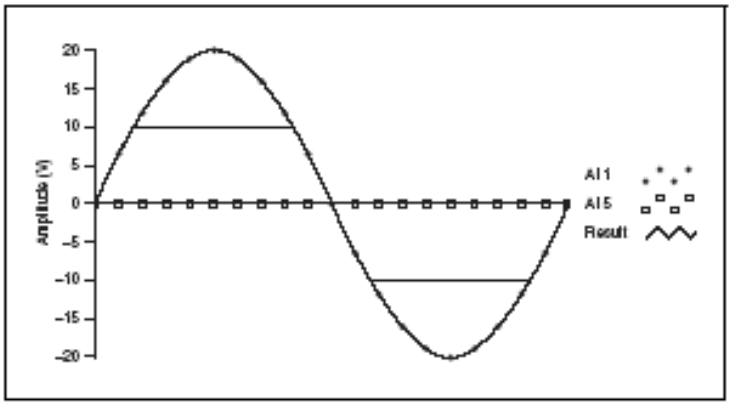


Figura 45. Salida recortada con señal más alta de ±10v

Si se conecta una señal más alta de ±10v en cualquier pin el resultado que se obtiene en la salida es recortada (Vea Figura 45).

Para poder conectar una señal en modo diferencial debe cumplir unos requisitos:

- La señal de entrada es de bajo nivel (menos de 1 voltio).

- Las puntas que conecta a la señal es mayor que 10pies.

- La señal de entrada requiere un punto de referencia a tierra separado o señal de retorno.

- Las puntas de la señal se mueven a través de un medio ruidoso.

- **MODO DE NODO SIMPLE**

Para conectar una señal de voltaje con referencia(RSE) a la tarjeta, se conecta el terminal positivo de la señal deseada al terminal AI y la tierra al terminal GND (Vea Figura 46). Esta fue la forma en la que se midió la entrada análoga.

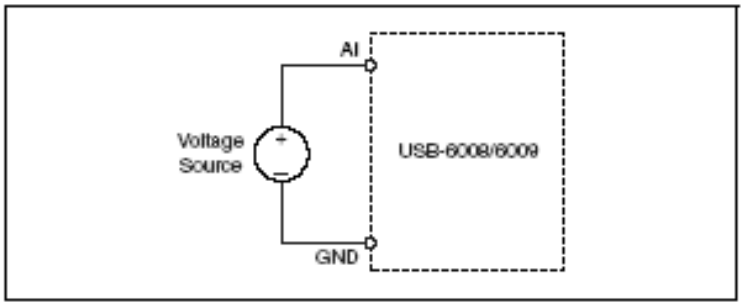


Figura 46. Esquema en modo de nodo simple

Para poder conectar una señal en modo nodo simple debe cumplir unos requisitos:

- La señal de entrada es de alto nivel (más de 1 voltio).

- Las puntas que conecta a la señal es menor que 10pies.

- La señal de entrada puede compartir un punto de referencia común con otras señales.

1.2.3 SALIDAS ANALOGAS

En la tarjeta USB 6008 se encuentran dos canales de salidas análogas que pueden generar una salida para 0-5v (Vea Figura 47).

Circuito:

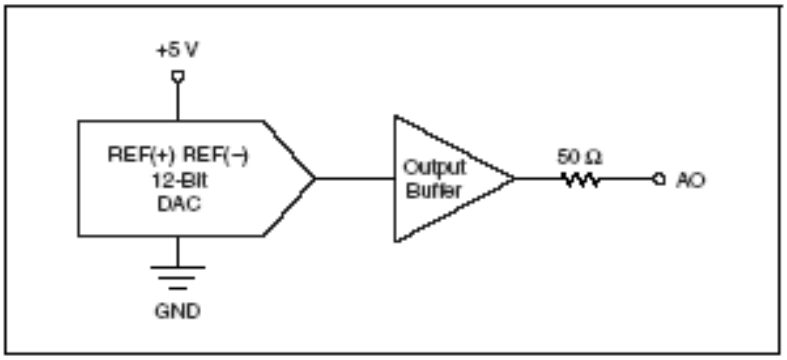


Figura 47. Diagrama de salidas análogas

- **DACs (convertidores de digital a análogo):**

Este convierte un código digital en voltajes análogos.

- **CONEXIÓN DE CARGA A LA SALIDA ANALOGA**

Para lograr conectar cargas a la tarjeta, se debe conectar el pin positivo de la carga al Terminal de la AO (salida digital), y conectar la tierra de la carga al terminal (Vea Figura 48).

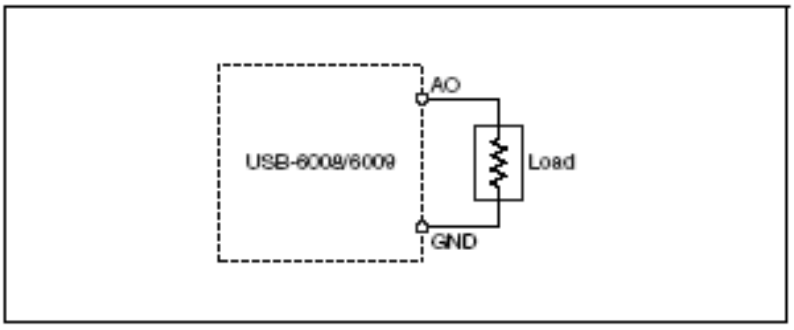


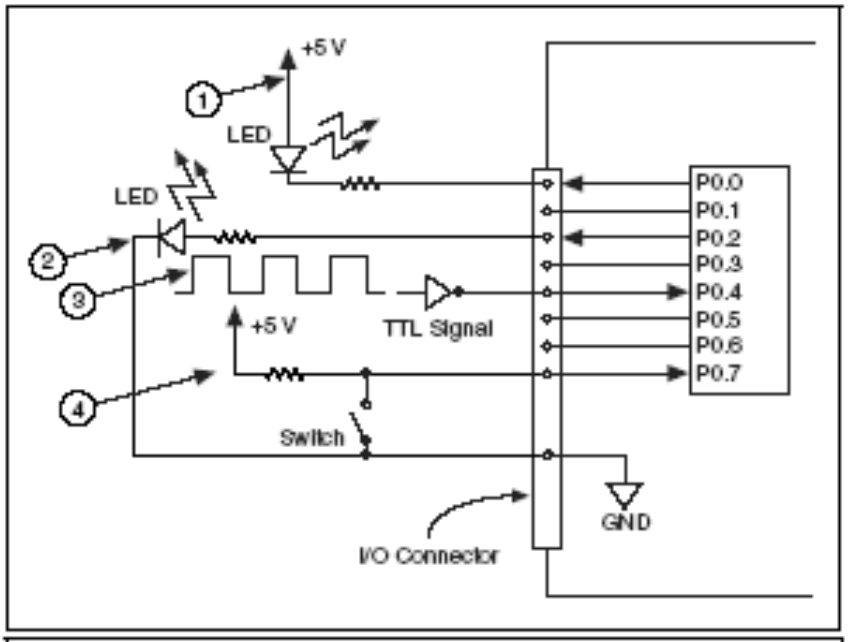
Figura 48. Conexión de carga a la salida análoga

1.2.4 ENTRADAS Y SALIDAS DIGITALES

La tarjeta contiene 12 líneas de digitales, PO<0….7>(puerto 0) y PI<0….3>(puerto 1), las cuales comprenden el puerto DIO(salidas /entradas digitales).Tierra es la referencia del puerto.

Las entradas o salidas digitales pueden tomarse como una línea individual o como un bus de datos, siendo este un conjunto de líneas o canales. Aunque no sea permitido configurar las líneas de un bus de datos individualmente como entrada o salida, si estas residen en un mismo puerto, todas las líneas de un bus de datos tienen la misma dirección. La Figura 49 muestra un ejemplo de conexión de carga a la salida análoga.

Circuito:



1 P0.0 configured as an open collector digital output driving a LED
2 P0.2 configured as a active drive digital output driving a LED
3 P0.4 configured as a digital input receiving a TTL signal from a gated inverter
4 P0.7 configured as a digital input receiving a 0 V or 5 V signal from a switch

Figura 49. Ejemplo de conexión de carga a la salida análoga

2. TOOLBOX DE ADQUISICION DE DATOS

Antes de emprender cualquier sistema de adquisición de datos se debe tener claro lo que se quiere medir, las características físicas de las mediciones, el sensor apropiado que se usa y el hardware o equipo apropiado para la adquisición de datos.

La Toolbox de Adquisición de Datos es una colección de funciones de archivos .m y archivos .mex construido sobre el entorno de MATLAB. La Toolbox también incluye varios vínculos de librerías dinámicas (DLL's) llamados adaptadores, los cuales permiten interactuar con un hardware específico. La Toolbox tiene las siguientes características:

- Un marco para traer datos reales, medidos en el espacio de trabajo de MATLAB usando un hardware de adquisición de datos compatible con el PC.

- Soporte para entradas análogas (AI), salidas análogas (AO) y entradas y salidas digitales (DIO). Incluyendo conversiones simultáneas para entradas y salidas análogas.

- Soporte para estos tipos de hardware más populares:

- Advantech®. Tarjetas que usen el administrador de dispositivos de Advantech.

- Módulos VXI E1432A/33A/34A de Agilent Technologies®.

- Keithley®. Tarjetas que usen DriverLINX.
- Tarjetas de Measurement Computing™ Corporation.

- National Instruments®. Tarjetas que usan el software tradicional NI-DAQ o NI-DAQmx.

- Puertos paralelos LPT1 y LPT3.

- Microsoft® Windows®. Tarjetas de sonido.
- Adicionalmente, se puede usar un Kit de adaptador de interface para dispositivos que no soportan la Toolbox.

- Eventos de adquisiciones dirigidas.

2.1 PREREQUISITOS

Para adquirir datos medidos en el área de trabajo de Matlab o sacar datos desde la misma es necesario instalar los siguientes componentes:

- MATLAB

- La “Toolbox” de adquisición de datos

- Soporte del dispositivo de adquisición de datos

- Software - tales como controladores y librerías

2.2 VERIFICACION DE LA TOOLBOX INSTALADA.

Para determinar si la “Toolbox” de Adquisición de Datos está instalada en el sistema se usa el comando:

daqhwinfo

ver proporciona una lista de los componentes de Matlab instalados, el número de licencia de Matlab y la versión de cada componente.

2.3 COMPONENTES DE LA TOOLBOX

La “Toolbox” de Adquisición de Datos consta de tres componentes:

- Funciones de archivos m.

- Motor de adquisición de datos.

- Adaptadores de Hardware.

Como se muestra en la Figura 50, estos componentes permiten pasar información entre MATLAB y el hardware de adquisición de datos.

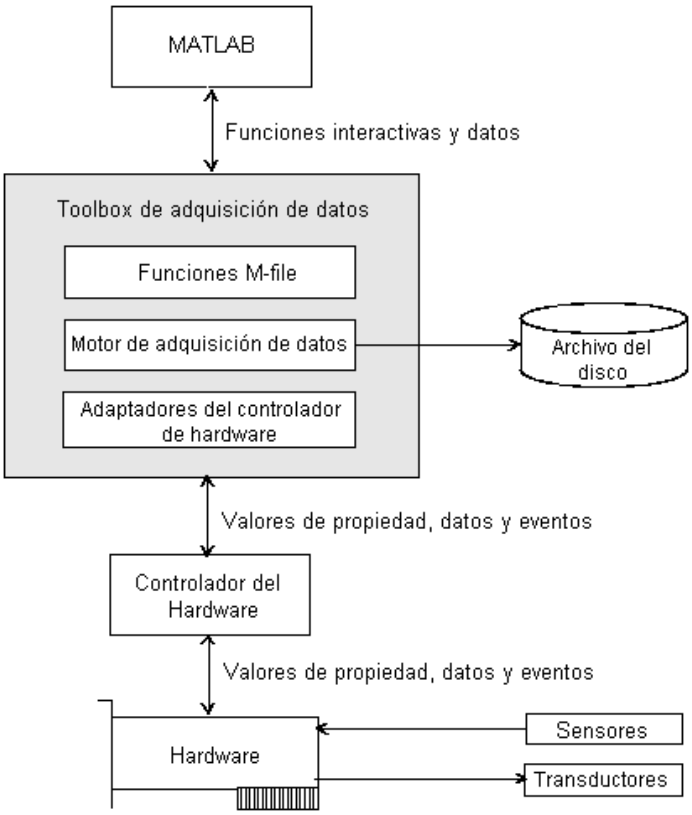


Figura 50. Componentes de la toolbox

- **Valores de Propiedad:**

Es posible controlar el comportamiento de la aplicación de adquisición de datos, configurando los valores de propiedad. En general, se puede pensar que una propiedad es como una característica de la Toolbox o del controlador del hardware que puede ser manipulado para satisfacer las necesidades de la aplicación.

- **Datos:**

Se puede adquirir datos desde un sensor conectado a un subsistema de entrada análoga y guardarla en MATLAB, o sacar datos de MATLAB a un transductor conectado a un subsistema de salida análoga. Adicionalmente puede transferir valores (1s y 0s) entre MATLAB y un subsistema de entrada y salida digital.

- **Eventos:**

Un evento ocurre en un instante particular después que una condición es cumplida y puede resultar en uno o más llamados que estén especificados. Los eventos pueden ser generados sólo después que se configure las propiedades asociadas. Algunas de las maneras en las que se puede usar los eventos, incluye la iniciación del análisis después que una predeterminada cantidad de datos sea adquirida, o mostrando un mensaje al espacio de trabajo de MATLAB después que un error ocurra.

2.3.1 FUNCIONES DE ARCHIVOS.M

MATLAB permite crear funciones propias en forma de archivos .m. Un archivo .m de función es similar a un archivo script, al igual que ellos son archivos de texto creados en un editor de texto. La diferencia entre ambos es que la función sólo se comunica con el espacio de trabajo a través de las variables de entrada y salida, las variables intermedias dentro de la función no aparecen ni interactúan con el espacio de trabajo de MATLAB.

2.3.2 MOTOR DE ADQUISICION DE DATOS

El motor de la adquisición de datos es un archivo MEX que:

- Almacena los objetos de dispositivo y valores de propiedades asociadas que controla la adquisición de datos.

- Controla la sincronización de eventos.

- Controla el almacenamiento de los datos adquiridos o de los datos en cola.

Mientras que el motor realiza esas tareas, se puede realizar otras tareas en MATLAB como analizar los datos adquiridos. En otras palabras, el motor y MATLAB son asíncronos. La relación entre adquirir datos, sacar datos y el flujo de datos es descrito a continuación.

2.3.3 ADAPTADORES DE HARDWARE

El adaptador es la interfaz entre el motor de adquisición de datos y el controlador del hardware.

El propósito principal del adaptador es pasar información entre MATLAB y el hardware por medio de sus controladores.

Para adquirir datos usando una tarjeta de National Instruments, la versión apropiada del driver NI-DAQ debe ser instalada en su plataforma.

En un sistema de adquisición de datos, el principal objetivo que tiene es el de proporcionar las herramientas y recursos necesarios para tomar señales físicas y convertirlas en datos que posteriormente se puedan procesar y mostrar.

Un sistema de adquisición de datos se podría tomar como un grupo de hardware y software que permiten interactuar con el mundo real, consta de estos componentes:

2.4 HARDWARE DE ADQUISICION

Es el corazón o motor de cualquier sistema de adquisición de datos. Su función principal es hacer la conversión de señales análogas a señales digitales y señales digitales a análogas. Conversión A/D y D/A.

El hardware de adquisición de datos puede instalarse de manera interna, directamente en una ranura de expansión del computador o de manera externa, que se conecta al computador a través de un cable.

Se caracteriza por los subsistemas que éste posee. Un subsistema es un componente del hardware que realiza una tarea específica. Los más comunes son:

- Entradas análogas
- Salidas análogas
- Entradas/salidas digitales
- Contador/Temporizador

2.5 ACONDICIONAMIENTO DE SEÑAL

- **Sensores y Actuadores (Transductores):**

Un transductor es un dispositivo que convierte un tipo de energía de entrada en una energía de salida de otra forma.

- **Acondicionamiento de señal**

Las señales de los sensores con frecuencia son incompatibles con el hardware de adquisición de datos, por lo tanto deben ser acondicionadas para superar este tipo de inconveniente.

Por ejemplo, las señales podrían ser amplificadas o volverlas en señales sin componentes de frecuencias indeseadas. Las señales de salida también pueden ser acondicionadas.

El tipo de acondicionamiento depende del sensor que se utilice. Las formas más comunes de acondicionar una señal son:

- **Amplificación**

La amplificación permite reducir y hacer uso del mayor rango y así aumentar la resolución de la medición.

- **Filtrado**

El filtro de ruido es usado en señales que varían lentamente, como la temperatura. En cambio las señales de rápida variación como la vibración, requieren de un tipo de filtro diferente, conocido como filtro de antialiasing, que eliminan las frecuencias más altas, que podrían dar lugar a medidas erróneas.

- **Multiplexación**

Permite enviar distintas señales sobre un mismo canal. Se debe tener en cuenta que la señal de conmutación de un multiplexor tenga tiempo suficiente para realizar esto.

- **Computador**

Proporciona un procesador, un sistema de reloj, un bus de datos, memoria y espacio en el disco para almacenar datos.

- **Software**

Permite el intercambio de información entre el computador y el hardware. El software por lo general es NI-DAQ o NI-DAQmx (adaptador del hardware).

En la Figura 51 se presentan los componentes y la relación entre cada uno:

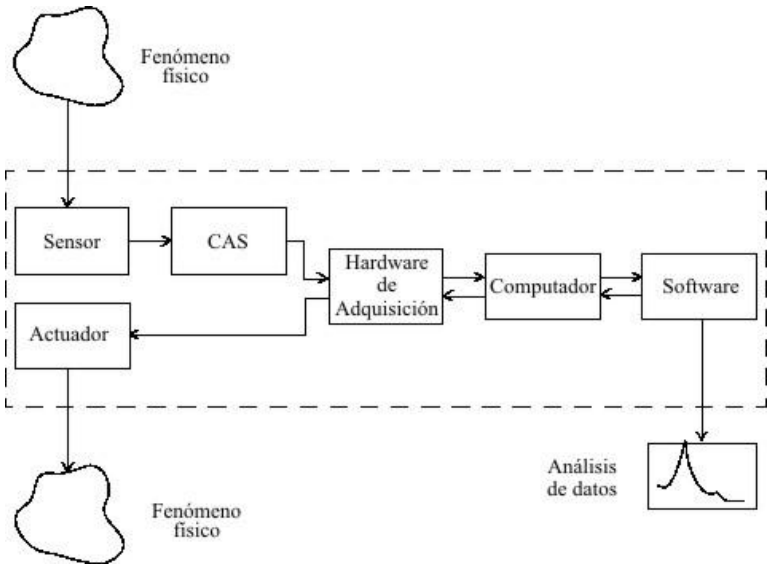


Figura 51. Componentes y la relacion entre cada uno

2.6 EXACTITUD Y PRESICIÒN

La exactitud de una medición determina qué tan cerca llegan a ser las mediciones al valor real. La precisión de la medición refleja que tan exacto es un resultado. Cada vez que se adquieren los datos medidos, se debe hacer todo lo posible para maximizar la exactitud y la precisión. La calidad de la medición depende de la exactitud y la precisión de todo el sistema de adquisición de datos, y puede ser limitada por factores tales como la resolución de la tarjeta o el ruido ambiental.

2.7 RUIDO

El ruido puede se generado dentro de los componentes eléctricos de la entrada del amplificador (ruido interno), o éste puede ser adherido a la señal como mientras es transferida por los cables de entrada (ruido externo).

El ruido se añade al circuito de adquisición desde esas fuentes externas, porque las puntas de la señal actúan como antenas que toman actividad eléctrica del ambiente.
Gran parte de este ruido es común a ambos cables de señal. Para eliminar la mayor parte de este voltaje en modo común, se debe:

- Configurar los canales de entrada en modo diferencial.

- Utilice cables de señal que son trenzados.

- Mantenga los cables de señal lo más corto posible.

2.8 FLUJO DE DATOS ADQUIRIDOS

En el flujo de datos, los datos son almacenados temporalmente en la memoria porque estos se van sobrescribiendo. La tasa con la cual los datos son sobrescritos depende de factores incluyendo la memoria disponible, la tasa con la que los datos son adquiridos y el número de canales del hardware.

Los datos almacenados no están disponibles automáticamente en el área de trabajo de MATLAB. Por ello, se tiene que extraer explícitamente del motor usando la función *getdata.* El flujo de datos adquiridos consiste de dos pasos independientes (Vea Figura 52):

1. Los datos adquiridos desde el hardware son almacenados en el motor.

2. Los datos son extraídos desde el motor y almacenados en el espacio de trabajo de MATLAB o sacados como archivo.

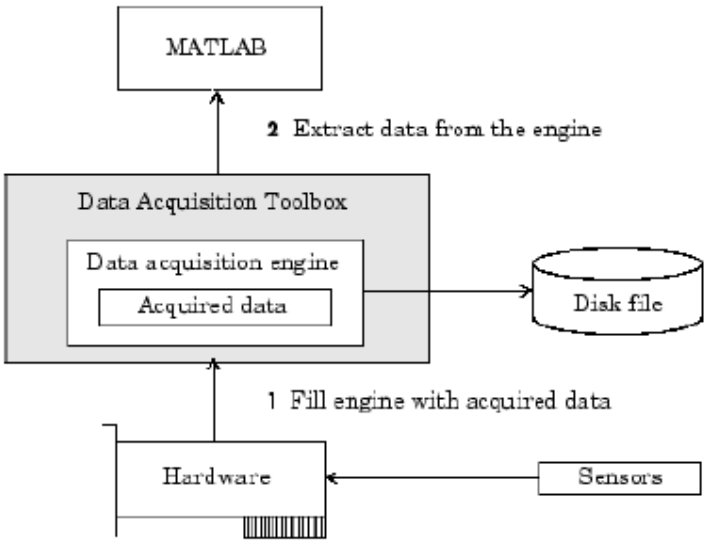


Figura 52. Pasos en el flujo de datos adquiridos

2.9 FLUJO DE SALIDA DE DATOS

El flujo de salida de datos se refiere al flujo de datos desde el motor de adquisición de datos al hardware. Sin embargo, antes que los datos son sacados, se deben poner en cola en el motor con la función *PUTDATA.* La cantidad de datos que se puede poner en cola depende de factores incluyendo la memoria disponible, el número de canales del hardware y el tamaño de cada muestra de dato (Vea figura 53).

El flujo de salida de datos consiste de dos pasos independientes:

- Los datos son puestos en cola en el motor desde el espacio de trabajo de MATLAB.

- Los datos en cola en el motor son sacados al hardware.

Estos pasos son ilustrados a continuación:

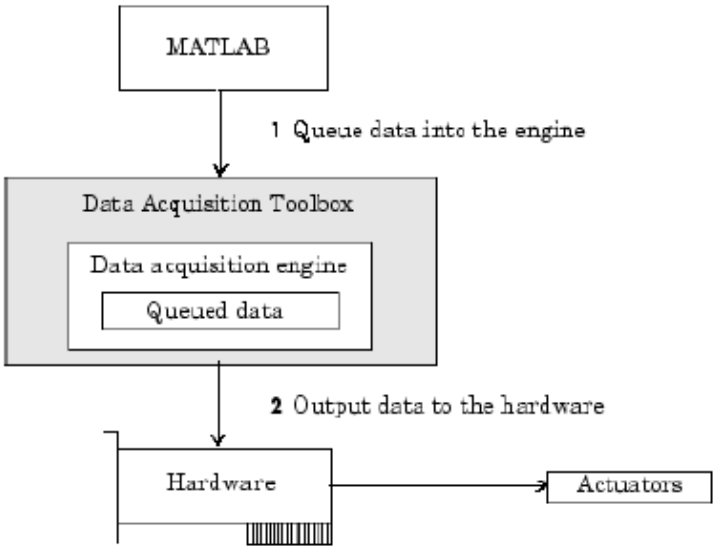


Figura 53. Pasos en el flujo de salida de datos

2.10 TASA DE MUESTREO ADECUADA PARA MUESTREAR UNA SEÑAL

Cuando se muestrea una señal continua, gran parte de esa información muestreada se pierde. El objetivo principal para que esto no suceda, es lograr muestrear a una tasa determinada buscando que la señal de interés este bien caracterizada, teniendo como resultado que la cantidad de información que se llega a perder sea mínima.

Hay factores que se deben tener en cuenta si se habla de la tasa de muestreo. Si se muestrea a una tasa muy baja puede ocurrir el efecto aliasing. El efecto aliasing se genera cuando la señal muestreada contiene componentes de frecuencias mayores a la mitad de la tasa de muestreo. Las reglas usadas para prevenir el antialiasing está dado por el Teorema de Nyquist, que establece que:

- Una señal análoga puede ser reconstruida sin errores únicamente desde las muestras tomadas con un intervalo de tiempo igual.

- La tasa de muestreo puede ser igual o dos veces más grande que el mayor componente de frecuencia de la señal análoga. Una frecuencia de la mitad de la tasa de muestreo es llamada como frecuencia de Nyquist.

3. EJEMPLOS

→EJEMPLO DE CONFIGURACIÓN DE SALIDA DIGITAL COMO LINEA INDIVIDUAL.

Para configurar una salida digital como línea individual se debe tomar tan sólo un ID de cualquier línea de un puerto.

Por ejemplo, si se quiere agregar al objeto DIO la línea tres del puerto dos como salida, con el nombre de Salida, se usa la forma 4, con los siguientes argumentos de entrada.

Objeto= DIO (Objeto creado anteriormente)
IDLínea = 3
Puerto= 1
Dirección= out
addline(DIO,0,3, 'out' , 'Salida');

→EJEMPLO 1 DE CONFIGURACIÓN DE SALIDA DIGITAL COMO BUS

A diferencia de la salida individual, para configurar las salida digital como bus se le debe agregar al objeto dos o más líneas de cualquiera de los puertos de la tarjeta teniendo en cuenta que todas las líneas deben tener la misma dirección ('out').

Por ejemplo, para agregarle ocho líneas de salida desde el puerto 0 al objeto creado anteriormente (DIO).

Objeto= DIO
Líneas= 0:7 (Desde la 0 hasta la 7)
Puerto= 0
Dirección= out

Para este caso no es necesario especificar el puerto puesto que al tomar ocho líneas, las toma del puerto 0, puesto que este es el número de líneas que tiene disponible.

→EJEMPLO 2 DE CONFIGURACIÓN DE SALIDA DIGITAL COMO BUS

Agregar 3 líneas de salida desde el puerto 1 al objeto, con nombres de linea1, linea2 y linea3.

Objeto= DIO
Líneas= 0:2 (Desde la 0 hasta la 2)
Puerto= 1
Nombres= linea1, linea2, linea3
Dirección= out
Addline(DIO,0:2,1,'out' ,{'linea1' , 'linea2' , 'linea3'})

En este ejemplo es necesario especificar el ID del puerto porque de no ser así, por defecto toma las 3 primeras líneas del puerto 0.

→EJEMPLO DE CONFIGURACIÓN DE ENTRADA DIGITAL COMO LINEA INDIVIDUAL.

Al ser una entrada digital individual, al objeto DIO se le agrega tan sólo una línea.
Para agregar una línea como salida a un objeto digital con el nombre Entrada se realiza de la forma 3, con los siguientes argumentos de entrada a la función ADDLINE.

Objeto= DIO
IDLínea = 0
Dirección= in
Nombre= Entrada
addline(DIO,0,'in','Entrada');

Cuando no se especifica el puerto, por defecto el objeto toma el puerto 0 y la línea que se agregue puede ser cualquiera de este puerto.

--> Se quiere agregar la línea 2 del puerto 1 a un objeto de entrada digital. Para esto se debe tomar la forma 2 con los siguientes argumentos de sintaxis.

Objeto= DIO
IDLínea = 2
Puerto= 1
Dirección= in

```
addline(DIO,2,1,'in');
```

→ EJEMPLO 1 DE CONFIGURACIÓN DE ENTRADA DIGITAL COMO BUS

Agregarle 3 líneas del puerto 0, con los nombres de Entrada1, Entrada2 y Entrada3, al objeto de entrada digital DIO.
Esto se realiza de la forma 4, teniendo como argumentos de entrada para la función:

Objeto= DIO (Objeto creado anteriormente)
IDLínea = 0:5
Puerto= 0
Dirección= in
Nombres de las líneas= Entrada1, Entrada2, Entrada3

```
addline(DIO,0:2,0,'in' ,{'Entrada1','Entrada2','Entrada3'} );
```

En este caso se puede o no, especificar el ID del puerto como argumento para la sintaxis, puesto que por defecto el objeto tomaría las tres primeras líneas del puerto 0.

-→EJEMPLO 2 DE CONFIGURACIÓN DE ENTRADA DIGITAL COMO BUS

Agregar 12 líneas, al objeto de entrada digital DIO.

En este ejemplo, es necesario hacer dos llamados a la función ADDLINE. El primer llamado para agregarle al objeto las 8 líneas del puerto 0 y el segundo para agregarle las 4 líneas del puerto 1.

LLAMADO 1	LLAMADO 2
Objeto= DIO	Objeto= DIO
IDLínea = 0:7	IDLínea = 0:3
Puerto= 0	Puerto= 1
Dirección= in	Dirección= in

```
addline(DIO,0:7,0,'in');
addline(DIO,0:3,1,'in');
```

→EJEMPLO DE CONFIGURACIÓN DE ENTRADA DIGITAL COMO BUS (EXTRAER DATOS).

Se tiene un objeto de ocho líneas de entrada digital, y se quiere extraer los valores de las líneas 0 y 1 de la tarjeta.

```
Datos = getvalue(DIO.Line(1:2));
```

En este caso se debe especificar los índices de la línea o de las líneas a las cuales se les quiere extraer los valores. Los índices utilizados son los de la nomenclatura de MATLAB.

Como los valores de un objeto digital en bus son extraídos como un vector binario (binvec), se pueden convertir en decimales con la función BINVEC2DEC.

Una vez creado el objeto digital con sus respectivas líneas de entrada, el paso a seguir es extraer los datos y guardarlos en una matriz de i filas por n columnas, donde i es el número de veces que se va a extraer datos y n es el número de líneas contenidas por el objeto.

Esto se crea dentro de un ciclo FOR.

```
a = 20;
for i=1:a
valores = getvalue(DIO)
datos(i,:)=binvec2dec(valores);
pause(1)
End
Datos
```

a: Cantidad de veces que se extrae datos.

Valores: Toma los valores de DIO en cada iteración.

Datos (i, :): Guarda en matriz de i x n los valores extraídos en cada iteración, donde n es el número de canales contenidos por el objeto.

Datos: muestra todos los datos leídos, en decimales.

→EJEMPLO DE CONFIGURACIÓN DE SALIDA ANÁLOGA COMO BUS.

La tarjeta cuenta con dos terminales de salida análoga, por lo tanto, para tomar una salida en bus, ésta debe contener los dos canales.

Agregar los canales 0 y 1 a un objeto de salida análoga, con los nombres de Salida1 y Salida2, estableciendo manualmente los índices de MATLAB.

```
Objeto = AO
Idcanal = 0:1
Indice = [1 2]
Nombres = {'Salida1','Salida2'}

Chan = addchannel(AO,0:1,[1 2],{'Salida1' , 'Salida2'})
```

Aunque los índices de MATLAB son establecidos automáticamente al agregar canales a un objeto, también se puede hacer manualmente.

→EJEMPLO 1 DE ENTRADA DIGITAL

En este ejemplo se crea un objeto digital DIO y se le agrega la línea 5 del puerto 0 con dirección de entrada, luego se crea un ciclo repetitivo, para extraer 10 veces el valor que hay en la entrada y se guarda todos los valores en una variable.

```
DIO = digitalio('nidaq','Dev1');
línea = addline(DIO,5,0,'in');
n=2;
a=10;
for i=1:a
valores(i,:) = getvalue(DIO);
pause(n)
end
delete (DIO)
clear DIO
```

→EJEMPLO 2 DE ENTRADA DIGITAL

En este ejemplo se tiene un objeto digital (DIO), al cual se le agrega las líneas del 0 al 3 del puerto 0 como salidas y la línea 0 del puerto 1 como entrada (la línea toma el índice 5 de MATLAB).

Mientras el valor en la línea de entrada es 1, se realiza un conteo regresivo del 9 al 0 con las líneas de salida conectadas al decodificador del display 7 segmentos de cátodo común.

```
DIO=digitalio('nidaq','Dev1');
Salidas=addline(DIO,0:3,'out');
Entrada=addline(DIO,0,1,'in','entrada');
datos=[9 8 7 6 5 4 3 2 1 0];
while getvalue(DIO.line(5))==1
for i=1:length(datos)
putvalue(DIO.line(1:4),datos(i))
pause(1)
End
End
delete (DIO)
clear DIO
```

→EJEMPLO 1 DE SALIDA DIGITAL

Este ejemplo ilustra los pasos básicos para escribir en un objeto digital (DIO) de salida individual, dos valores digitales con un tiempo de espera de 2 segundos entre ellos.

```
DIO=digitalio('nidaq','Dev1');
Linea=addline(DIO,0,'out');
n=2;
putvalue(DIO,0)
pause(n)
putvalue(DIO,1)
delete(DIO)
clear DIO
```

→EJEMPLO 2 DE SALIDA DIGITAL

En este ejemplo se tiene un arreglo de decimales (conteo del 0 al 15) que son escritos en un objeto digital (DIO) de salida en bus, que contiene las líneas del 0 al 3 del puerto 1 de la tarjeta.

```
DIO=digitalio('nidaq','Dev1');
Lineas=addline(DIO,0:3,1,'out');
n=1.5;
datos=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
for i=1:length(datos)
putvalue(DIO,datos(i))
pause(n)
end
delete(DIO)
clear DIO
```

→EJEMPLO 1 DE ENTRADA ANÁLOGA

Se crea un objeto de entrada análoga (AI), con un canal de nodo simple, para muestrear durante un segundo la salida del generador de onda seno (F=90Hz) del módulo de pruebas, a una tasa de muestreo de 1000Hz y se extraen 200 muestras del motor de adquisición para visualizarlas.

```
AI = analoginput('nidaq','Dev1');
Canal=addchannel(AI,0);


duracion=1;
set(AI,'InputType','SingleEnded')
set(AI,'SampleRate',1000)
TasaActual=get(AI,'SampleRate');
set(AI,'SamplesPerTrigger',TasaActual*duracion)


start(AI)
```

```
wait(AI,duracion+1)

[data,t] = getdata(AI,200);
figura
plot(t,data),
grid on

delete(AI)
clear AI
```

→EJEMPLO 1 DE SALIDA ANÁLOGA

Se crea un objeto de salida análoga AO de un canal, para sacar una señal seno de 0 a 5Voltios, creada con la función LINSPACE, como un vector fila de 100 valores desde 0 hasta 2*pi.

```
AO = analogoutput('nidaq','Dev1');
Canal = addchannel(AO,0);
datos = 2.5*sin(linspace(0,2*pi,100))+2.5;
pause(1)
for i=1:length(datos)
putsample(AO,datos(i))
end
delete (AO)
clear AO
```

→EJEMPLO 1 DE ENTRADA Y SALIDA ANÁLOGA

Se crea un objeto de entrada análoga ai y salida análoga ao, para poder enviar durante 10 segundos, con una tasa de muestreo de 150 hz, una señal sinusoidal de 100 periodos, osea 10hz, con una amplitud de 2.5. Luego se recibe la señal que genera la planta a la que se le aplica la sinusoidal, la tasa de muestreo del puerto de entrada análoga se configura a 1khz, además se tiene en cuenta el

modo de medición que es de entrada modo simple debido a que la señal que estamos midiendo esta referenciada a tierra. La acción de extraer datos se hace muestra a muestra, osea se envía un dato y se recibe uno, para así poder enviar y recibir en tiempo real al mismo tiempo.

```
ao = analogoutput('nidaq','Dev2');
addchannel(ao,0);
ai = analoginput('nidaq','Dev2');
addchannel(ai,0);

set(ai,'inputType','singleEnded')
set(ai,'samplerate',1000);

d=10;
duracion=d;
set(ao,'samplerate',150)
actualrate=get(ao,'samplerate');
lens=actualrate*duracion;
pasos=lens-1;
f=100;

datos =2.5*sin(linspace(0,2*pi*f,lens))+2.5;

for i=1:length(datos)
putsample(ao,datos(i))
mode(i)=getsample(ai);
end
delete(ao)
clear ao
delete(ai)
```

→EJEMPLO 1 DE ENTRADA ANÁLOGA Y SALIDA DIGITAL

Se crea un objeto de entrada análoga ao y salida digital DIO, se envía el dato digital con el comando putvalue, y se establece el tiempo de cambio entre 0 y 1, para poder enviar el escalón.

Luego se debe recibir la respuesta del sistema a la entrada escalón, pero primero se debe configurar el modo de medición, la tasa de muestreo del puerto de entrada análoga, después se inicia el puerto con la función start para así poder extraer los datos con la función getdata.

```
DIO=digitalio('nidaq','Dev2');
línea=addline(DIO,0,'out');
ai = analoginput('nidaq','Dev2');
addchannel(ai,0);
set(ai,'inputType','singleEnded')
set(ai,'samplerate',333);% 2.4segundos se estabiliza
n=10;
putvalue(DIO,0)
pause(n)
putvalue(DIO,1)
start(ai)
mode2=getdata(ai);
```

ANEXO 2

1. INTERPOLACION LINEAL

La interpolación lineal es un método matemático para aproximar el valor de un punto.

Sean dos puntos (x0,y0) e (x1,y1) la ecuación de la recta que pasa por dichos puntos es:

$$\frac{x - x_0}{x_1 - x_0} = \frac{y - y_0}{y_1 - y_0}$$

Si se quiere hallar un valor de y, dado una x que esté entre x0 y x1 (x0<x<x1), se reorganiza la anterior y queda la ecuación que se debe utilizar:

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

2. TIEMPO DE ESTABLECIMIENTO DE UN SISTEMA

Para poder conocer el tiempo de establecimiento de un sistema primero se aplica a su entrada una señal tipo escalón la cual permite conocer la respuesta del sistema frente a cambios abruptos en su entrada. Así mismo permite obtener el tiempo de establecimiento de la señal, es decir el tiempo que se tarda el sistema hasta alcanzar el estado estacionario (Vea Figura 54).

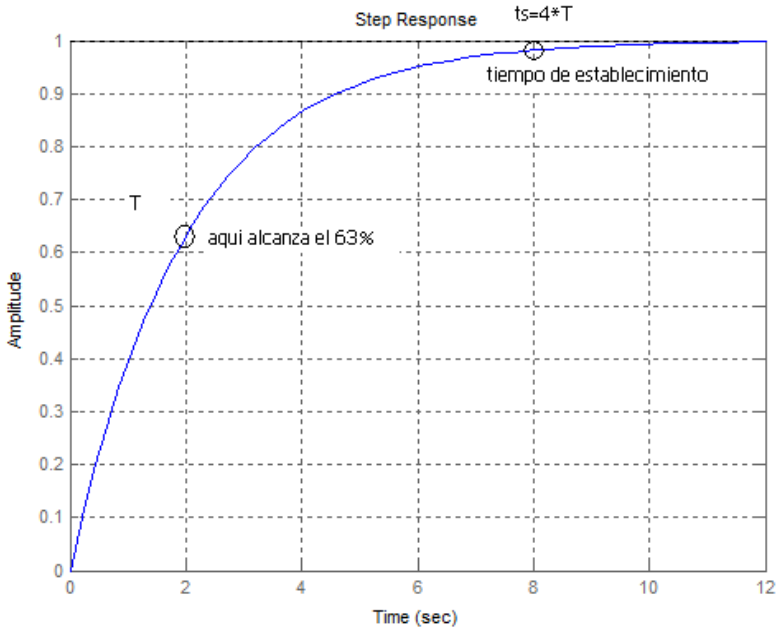


Figura 54. Tiempo de establecimiento de un sistema

Tomando como ejemplo un sistema de primer orden, se puede analizar el sistema de la siguiente manera:

$$G(s) = \frac{1}{2s + 1}$$

El numerador es la ganancia del estado estacionario, el valor de T osea 2, es el valor del tiempo en el cual el sistema alcanza aproximadamente el 63% de la respuesta. Teniendo T que es la constante de tiempo del sistema, se puede calcular el tiempo de establecimiento que es aproximadamente 4 veces esa constante de tiempo.

3. FILTROS

Se le llama filtrado al proceso mediante el cual se modifica una señal determinada de tal manera que las amplitudes relativas de las componentes en frecuencia cambian o incluso son eliminadas. Dicho de otra manera: un filtro es un dispositivo que impide o permite el paso de una cierta gama de frecuencias, donde permitir o impedir está relacionado con un nivel de atenuación o ganancia.
También sirven para restaurar una señal, cuando haya una señal que haya sido deformada de alguna forma.

Los filtros son sumamente importantes en sistemas de comunicaciones al igual que en control o telemetría entre algunas aplicaciones. Los filtros se pueden realizar de manera analógica o digital en configuraciones distintas tales que rechacen bandas de bajas frecuencias, altas frecuencias, frecuencias intermedias o alguna combinación de las anteriores. El dibujo de la Figura 55 muestra la idea básica de los filtros:

Figura 55. Funcionamiento externo de un filtro

Actualmente hay dos tipos de filtros, analógico y digital. Ambos son muy distintos en su construcción y en la forma en que tratan la señal. Los filtros analógicos son más baratos, rápidos y tienen un gran rango dinámico tanto en amplitud como en frecuencia. En cambio, los filtros digitales son enormemente superiores en el nivel de cumplimiento que los analógicos.

Otra de sus principales funciones de los filtros es la de prevenir el aliasing(este fenómeno sucede cuando se muestrea una señal con un valor inferior al

establecido en el Teorema de Nyquist: la frecuencia de muestreo mínima debe ser al menos el doble de la máxima frecuencia de la señal).

3.1 FILTRO ANALÓGICO.

Un filtro analógico se utiliza circuitos electrónicos que hacen uso de resistencias, condensadores y amplificadores para producir el efecto requerido en el filtrado.
Como circuitos de filtrado son comúnmente utilizados en aplicaciones de reducción de ruido, tratado de la señal de video, ecualizadores gráficos en los sistemas Hi-Fi, y en otras áreas.
En todas las etapas, la señal que se filtra es un voltaje eléctrico o corriente como puede ser la analógica directa de una cantidad física (un sonido, una señal de video) compleja.

3.2 FILTRO DIGITAL

El filtro digital es un sistema de tiempo discreto que puede realizar funciones de filtrado de señales. Un filtro digital requiere un procesador digital para realizar cálculos numéricos en los valores muestreados de la señal. El procesador puede ser un ordenador corriente, como un PC, o un chip DSP (Digital Signal Processor) especializado. La Figura 56 muestra el funcionamiento interno básico de un filtro.

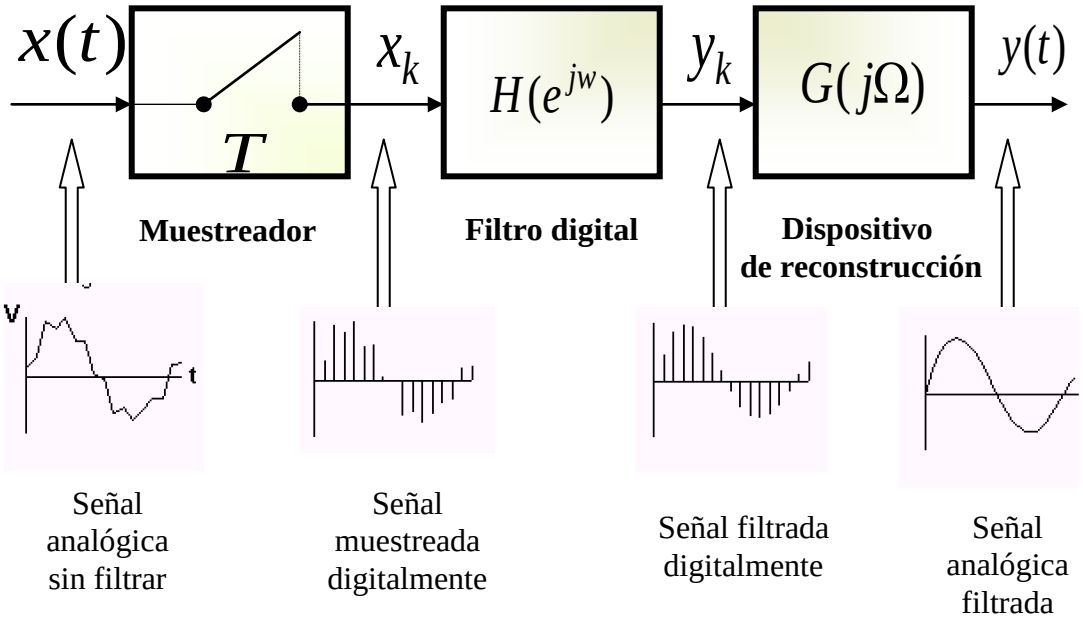


Figura 56. Funcionamiento interno básico de un filtro.

- **VENTAJAS DEL FILTRO DIGITAL**

1. Un filtro digital es programable, su función está determinado por un programa almacenado en el procesador. Esto significa que el efecto del filtro puede ser cambiado fácilmente sin modificar su circuitería (hardware). Un filtro analógico sólo puede cambiar rediseñando el circuito de filtrado.

2. Los filtros digitales son fácilmente diseñados, testados e implementados en un ordenador convencional

3. Las características de los circuitos de filtrado analógico (particularmente aquellos que contengan componentes activos) son susceptibles a las variaciones de velocidad y de temperatura. En cambio, los filtros digitales no sufren este problema, y son extremadamente estables con respecto al tiempo y la temperatura.

4. los filtros digitales sólo pueden tratar señales de baja frecuencia con gran exactitud. A medida que la velocidad de la tecnología DSP aumente, los filtros digitales podrán empezar a poderse aplicar en señales de alta frecuencia en el dominio de las frecuencias de radio, el cual fue un campo exclusivo reservado a la tecnología analógica.

5. Los filtros digitales son mucho más versátiles en su capacidad de procesar señales de diferentes formas. Esto significa que algunos filtros digitales tienen la capacidad de adaptarse a los cambios en las características de la señal.

6. Alta inmunidad al ruido.

7. Alta precisión (limitada por los errores de redondeo en la aritmética empleada).

8. Muy bajo coste (y bajando).

- **ORDEN DE UN FILTRO DIGITAL**

El orden de un filtro digital es el numero de las entradas *anteriores* (almacenadas en la memoria del procesador) utilizadas para calcular la salida de la señal actual.

Orden cero: $y_n = a_0 x_n$

Primer orden: $y_n = a_0 x_n + a_1 x_{n-1}$

Segundo orden: $y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$

3.3 CLASIFICACION DE LOS FILTROS

La Tabla 3 resume la clasificación de los filtros digitales por su uso y por su implementación. El uso de un filtro digital puede estar en tres categorías: dominio del tiempo, dominio de la frecuencia y personalizados. Como han sido descritos anteriormente los filtros de dominio del tiempo son utilizados cuando la información está codificada en la forma de onda de la señal.

El filtrado del dominio del tiempo es usado en acciones tales como: suavizado, supresión DC, formado de ondulación, etc. Mientras que los filtros de dominio de la frecuencia son usados cuando la información se encuentra en la amplitud, la frecuencia y la fase de la componente sinuidal. El objetivo de este filtro es la de separar una banda de frecuencias de otra.

Los filtros personalizados son usados cuando se requiere una acción especial al filtro, son más elaborados que las cuatro respuestas básicas de (pasa-alto, pasa-bajo, pasa-banda y quita-banda).

Filtro usado para:	Filtro implementado por:	
	Convolution *Finite Impulse Reponse (FIR)*	Recursión *Infinite Impulse Reponse (IIR)*
Dominio del tiempo *(suavizado, supresión DC)*	Moving Average[1]	Polo simple
Dominio de la frecuencia *(separación de frecuencias)*	Windowed-Sinc	Chebishev
Personalizado *(Deconvolution)*	FIR personalizado	Diseño iterativo

Tabla 3. Clasificación de Filtros

3.3.1 FILTRO NO RECURSIVO (FIR)

Estos filtros son conocidos por las siglas FIR (Finite Impulse Response[2]), pues la respuesta de impulso es de una duración finita (finaliza cuando la señal de entrada finaliza).

- Tienen respuesta al impulso de duración finita.
- No tienen realimentación.

- Todos sus polos están en z=0, por tanto no tienen problemas de estabilidad.

- **MOVING AVERAGE (MEDIA MOVIL)**

El filtro moving average es el filtro más común en los DSP, en la mayoría de las veces se debe a ser el filtro digital más fácil de entender y usar. A pesar de su simplicidad, el filtro de la media móvil es óptimo para una tarea común como puede ser la reducción de un ruido fortuito mientras retiene una respuesta de paso muy bien definida.

Como su propio nombre indica, el filtro de la media móvil opera mediante el promedio de un número de puntos de la señal de entrada para producir cada punto de la señal de salida.
La media móvil es un mal filtro pasa-bajo, debido a una mala atenuación de la banda de parada.

- **WINDOWED SINC**

Los filtros que son windowed-sinc son utilizados para separar una banda de frecuencias de otra. Son muy estables, producen algunos efectos, y pueden ser forzados a diferentes niveles de rendimiento. Las características especiales del dominio de la frecuencia son obtenidas en detrimento de sus funcionalidades dentro del dominio del tiempo, incluyendo una ondulación excesiva y un *overshoot* en la respuesta de step. Este tipo de filtro es muy fácil de programar, pero tienen una ejecución muy lenta.

- **PERSONALIZADOS**

Muchos filtros sólo tienen una de las cuatro respuestas estándar de frecuencia: pasa-bajo, pasa-alto, pasa-banda o quita-banda. Se pueden diseñar filtros que con una respuesta de frecuencia arbitraria, se ajusten a las necesidades de una aplicación particular.

3.3.2 FILTRO RECURSIVO (IIR).

Un filtro recursivo es aquel que añade a los valores de entrada algún valor de salida previo. Estos, al igual que las entradas, son almacenados en la memoria del procesador.
Estos filtros son conocidos por las siglas IIR (Infinite Impulse Response), ya que su respuesta de impulso no finaliza porque los términos recursivos (las salidas anteriores) generan energía en la entrada del filtro y éste continúa funcionando. Realmente esta nomenclatura no es exacta, ya que en todos los filtros IIR la respuesta de impulso se reduce virtualmente a cero en un tiempo finito.

- Tienen respuesta al impulso de duración infinita.

- Tienen realimentación.

- Deben diseñarse con cuidado para evitar problemas de estabilidad.

- **BUTTERWORD**

Esta familia de filtros tiene buenas características transitorias.

La respuesta de frecuencia es extremadamente plana (Vea Figura 57), cerca a un nivel de DC y asintóticamente llega a una cuesta con una pendiente de n x 6 dB por octava en la región de la detección de la banda. En las proximidades de la frecuencia de corte es ligeramente redondeada.

La familia Butterworth es ampliamente utilizada, ya que en sus diseños se obtienen valores prácticos de los componentes con tolerancias menos críticas que los otros tipos de filtros.

Función de transferencia de los filtros Butterworth:

$$|H(j\omega)| = \frac{K_{Pb}}{\sqrt{1+\left(\omega/\omega_c\right)^{2n}}}$$

$$n = 1,2,3,\ldots$$

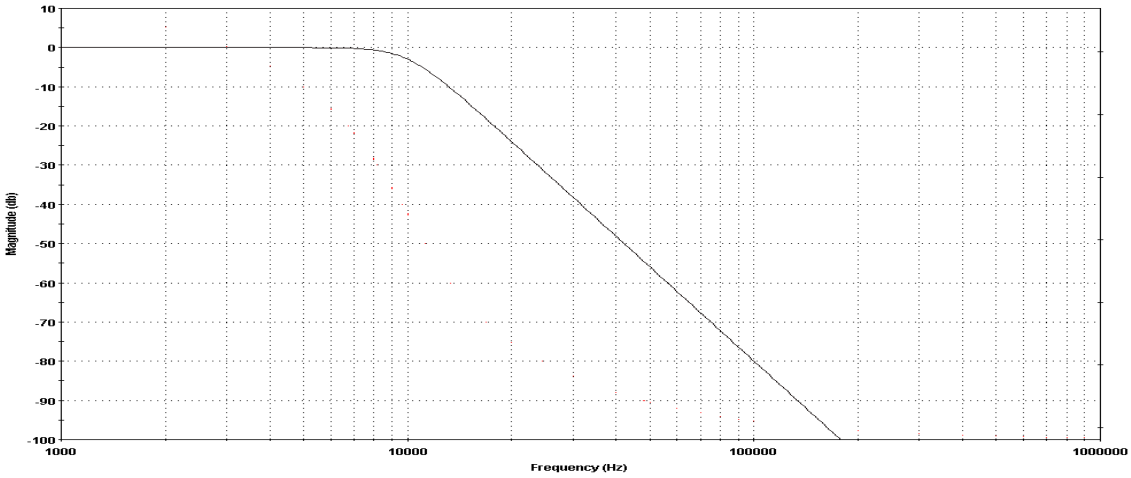


Figura 57. Respuesta en Frecuencia de un Filtro Butterworth fc=1kHz

3.4 COMPARACION ENTRE FILTROS IIR Y FIR

- Los filtros IIR producen en general distorsión de fase, es decir la fase no es lineal con la frecuencia.

- Los filtros FIR son de fase lineal.

- El orden de un filtro IIR es mucho menor que el de un filtro FIR para una misma aplicación.

- Los filtros FIR son siempre estables.

- Los filtros FIR tienen una mejor ejecución que los filtros IIR, pero su ejecución es mucho más lenta.

ANEXO 3

1. IDENTIFICACION DE UN SISTEMA:

Para poder aplicar las reglas de sintonización de controladores PI y PID lo primero que se debe hacer es la identificación del proceso, lo que quiere decir es que de acuerdo al modelo dinámico seleccionado se calculan las constantes y se obtiene la función de transferencia del proceso.

1.1 MODELOS DINAMICOS:

Los modelos dinámicos utilizados para la identificación de procesos son:

- **Primer orden**

$$Gp(s) = \frac{Kp}{\tau s + 1}$$

- **Primer orden más tiempo muerto**

$$Gp(s) = \frac{Kp\ e^{-t_m s}}{\tau s + 1}$$

- **Críticamente amortiguado más tiempo muerto (polo doble más tiempo muerto)**

$$Gp(s) = \frac{Kp\ e^{-t_m s}}{(\tau s + 1)^2}$$

- **Segundo orden sobre amortiguado**

$$Gp(s) = \frac{Kp}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

- **Segundo orden sub amortiguado**

$$Gp(s) = \frac{Kp\ \omega^2_n}{s^2 + 2\xi\omega_n s + \omega^2_n} = \frac{Kp}{\tau^2_n s^2 + 2\xi\tau_n s + 1}$$

- **Segundo orden sobre amortiguado más tiempo muerto**

$$Gp(s) = \frac{Kp\ e^{-t_m s}}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

- **Segundo orden sub amortiguado más tiempo muerto**

$$Gp(s) = \frac{Kp\ \omega^2_n e^{-t_m s}}{s^2 + 2\xi\omega_n s + \omega^2_n} = \frac{Kp\ e^{-t_m s}}{\tau^2_n s^2 + 2\xi\tau_n s + 1}$$

1.2 METODOS DE IDENTIFICACION (METODOS DE LAZO ABIERTO):

- **Métodos basados en la curva de reacción del proceso (o como se conoce respuesta al escalón)**

La curva de reacción del proceso se obtiene por medio de una prueba de lazo abierto con el controlador en manual y el sistema funcionando en el punto de operación deseado. En estas condiciones se aplica un cambio escalón en la salida del controlador y se registra esta señal y la de salida del proceso, desde el instante en que se aplicó el escalón de entrada hasta que el sistema alcance un nuevo punto de operación estable (Vea Figura 58).

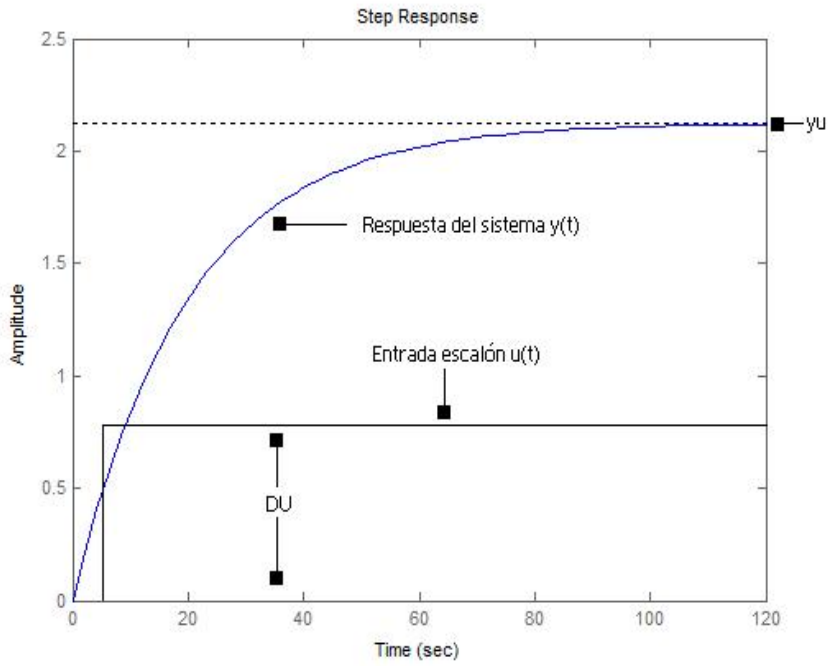


Figura 58. Respuesta al escalon

- **Para sistemas de segundo orden (o mayor) sobre amortiguados:**

Estos sistemas pueden ser aproximados mediante un modelo de primer orden más tiempo muerto. La gran mayoría de los procesos son identificados de esta manera, pero algunos necesitan modelarse por un sistema de segundo orden más tiempo muerto.

- **Métodos de la tangente (Vea Figura 59)**
  - Método de la tangente de ziegler y nichols
  - Método de identificación basado en la tangente, modificado de Murrill

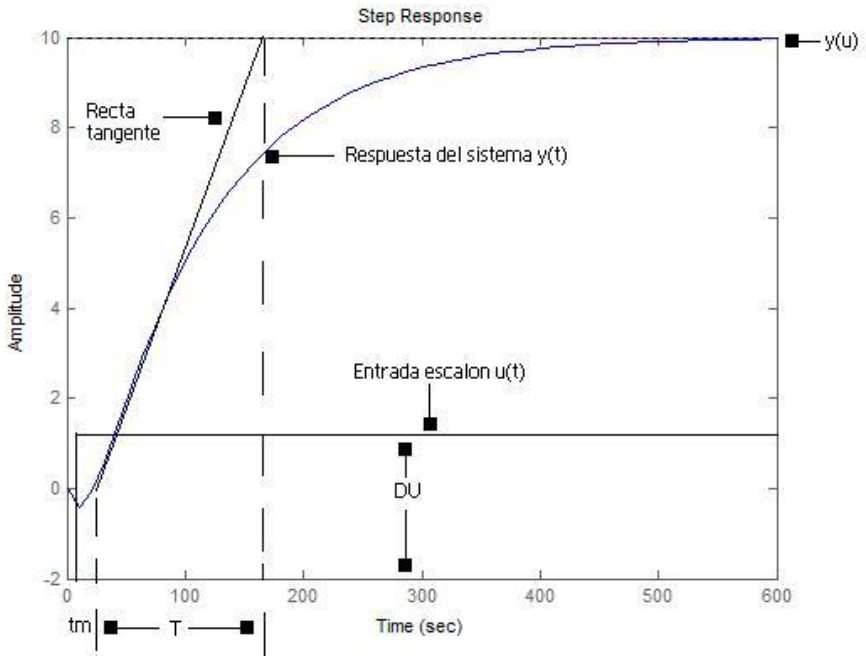


Figura 59. Metodo de la tangente para sistemas de segundo orden o mayor

- **Métodos de dos puntos.**

Los métodos que requieren el trazo de la recta tangente a la curva no siempre son fáciles de hacer, debido a que algunos pueden contener ruido y será difícil obtener el punto máximo de la pendiente, esto lleva a que la recta que se trazo no sea la ideal, afectando el tiempo muerto y la constante de tiempo del modelo.

Debido a esto se creó otro método utilizando 2 puntos de la curva de reacción (Vea Figura 60 y Tabla 4).

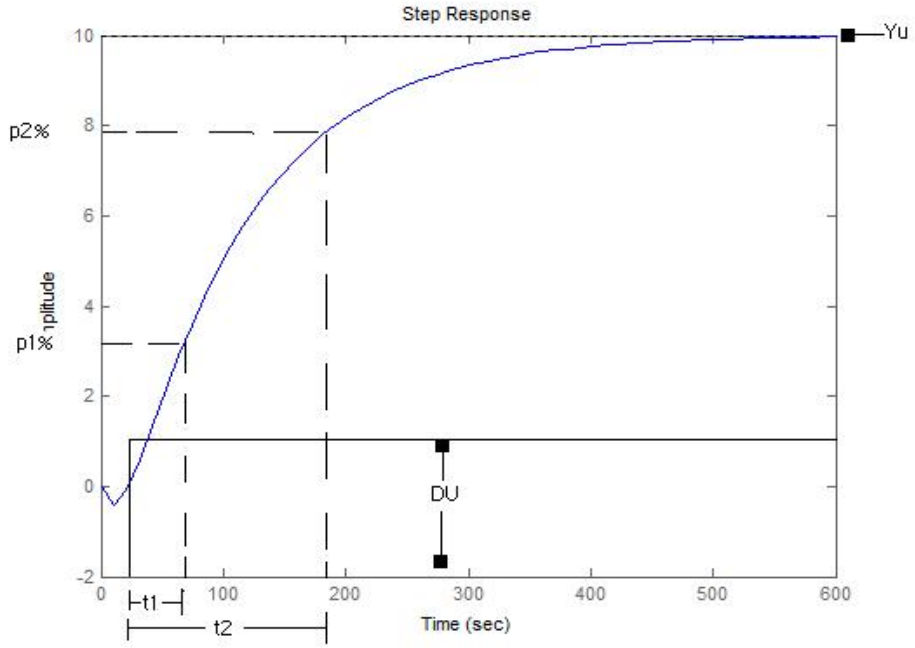


Figura 60. Método de dos puntos

Método	% p1(t1)	% p2(t2)	A	b
Alfaro	25	75	0.91	1.26
Broida	28	40	0.55	2.80
Ho	35	85	0.67	1.29
Chen y yang	33	67	0.14	1.54
Smith	28	63	0.15	1.50
Viteckova	33	70	1.24	1.49

Tabla 4. Constantes para la identificación de los modelos de primer orden más tiempo muerto

- **Modelo de primer orden más tiempo muerto:**

- **Método Smith**

$$y_u = Kp\Delta u$$

$$t1 = 28.3\% \quad \text{Y} \quad t2 = 63.2\%$$

$$\tau = a(t2 - t1)$$

$$t_m = bt_1 + (1 - b)t_2$$

- **Método de Alfaro**

$$Kp = y_u 7\Delta u$$

$$t1 = 25\% \quad \text{Y} \quad t2 = 75\%$$

$$\tau = a(t2 - t1)$$

$$t_m = bt_1 + (1 - b)t_2$$

- **Modelo de polo doble más tiempo muerto**

Método	% p1(t1)	% p2(t2)	A	b
Alfaro	25	75	0.578	1.555
Ho	35	85	0.463	1.574
Viteckova	33	70	0.794	1.937

Tabla 5. Constantes para la identificación

Se toman 2 puntos de la curva de reacción. Se usan para obtener un modelo de primer orden más tiempo muerto o de un modelo de polo doble más tiempo muerto. En este modelo de polo doble también se escogen 2 puntos porcentuales de la respuesta del sistema a un step.

Constantes para la identificación de los modelos de polo doble más tiempo muerto

$$Kp = y_u 7 \Delta u$$

$$\tau = a(t2 - t1)$$

$$t_m = bt_1 + (1 - b)t_2$$

- **Métodos de tres puntos (Vea Figura 61)**

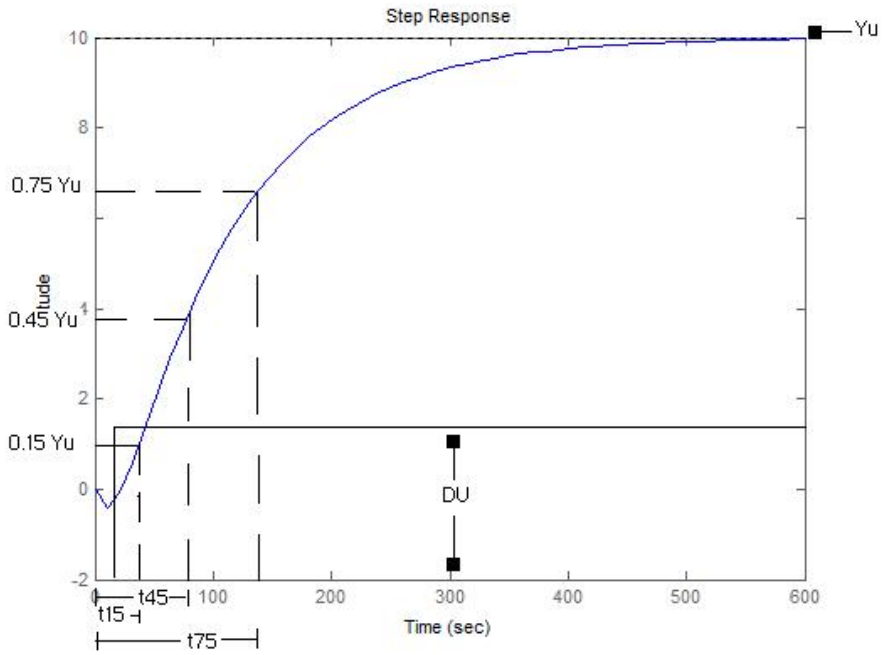


Figura 61. Método de tres puntos

Al igual que los métodos de dos puntos, este método no necesita del trazado de la recta tangente a la curva.

- **Modelo de la planta de segundo orden más tiempo muerto**

- **Método de Stark (sobre amortiguado)**

$$t1 = 15\%, t2 = 45\%\ y\ t3 = 75\%$$

$$Kp = \frac{y_u}{\Delta u}$$

$$x = \frac{t2 - t1}{t3 - t1}$$

$$\xi = \frac{0.0805 - 5.547(0.475 - x)^2}{x - 0.356}$$

$$f_2(\xi) = 2.6 * \xi - 0.60 \quad para\ \xi > 1$$

$$w_n = \frac{f_2(\xi)}{t3 - t1}$$

$$f_3(\xi) = 0.922(1.66)^{\xi}$$

$$t_{m=} t2 - \frac{f_3(\xi)}{w_n}$$

$$\tau_1 = \frac{\xi + \sqrt{\xi^2 - 1}}{w_n} \quad para\ \xi > 1$$

$$\tau_2 = \frac{\xi - \sqrt{\xi^2 - 1}}{w_n} \quad para\ \xi > 1$$

- **Para sistemas sub amortiguados**

Obtención de los parámetros mediante una observación directa de la curva de reacción del proceso de un sistema de segundo orden

- **Modelo de sistemas sub-amortiguados de segundo orden o mayor**

Cuando la respuesta del sistema a un cambio de tipo escalón en la señal de entrada, tiene un comportamiento sub-amortiguado, es erróneo utilizar modelos de primer orden o segundo orden con polos reales para su identificación. Es necesario utilizar un modelo que tenga polos complejos conjugados.

- **Método de Stark**

$$t1 = 15\%, t2 = 45\% \; y \; t3 = 75\%$$

$$Kp = \frac{y_u}{\Delta u}$$

$$x = \frac{t2 - t1}{t3 - t1}$$

$$\xi = \frac{0.0805 - 5.547(0.475 - x)^2}{x - 0.356}$$

$$f_2(\xi) = 0.708(2.811)^{\xi} \quad para \; \xi < 1$$

$$f_3(\xi) = 0.922(1.66)^{\xi}$$

$$w_n = \frac{f_2(\xi)}{t3 - t1}$$

$$t_m = t2 - \frac{f_3(\xi)}{w_n}$$

- **Método de identificación basado en el sobrepaso máximo (Vea Figura 62)**

Permite identificar sistemas sub-amortiguados de orden superior o igual a dos

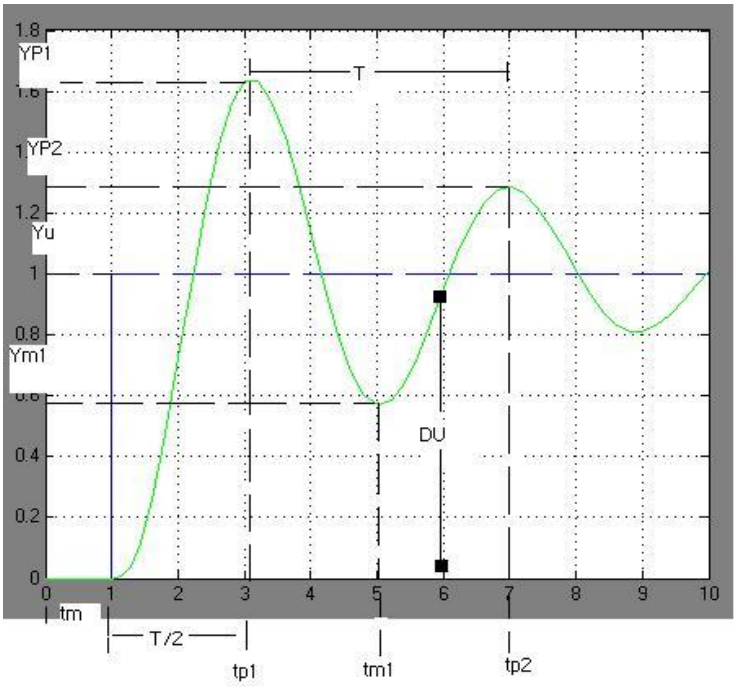


Figura 62. Metodo de identificacion basado en el sobrepaso máximo

$$Kp = \frac{(y_u - y_0)}{\Delta u}$$

$$\delta = \frac{y_{p1} - y_u}{y_u - y_0}$$

$$\xi = \sqrt{\frac{ln^2\delta}{\pi^2 + ln^2\delta}}$$

$$T = 2(t_{m1} - t_{p1})$$

$$w_n = \frac{2\pi}{T\sqrt{1-\xi^2}}$$

$$t_m = t_{p1} - \frac{T}{2}$$

- **Método de identificación basado en el decaimiento**

$$Kp = \frac{(y_u - y_0)}{\Delta u}$$

$$\varphi = \frac{y_{p2} - y_u}{y_{p1} - y_u}$$

$$\phi = \xi w_n = \frac{-\ln \varphi}{T}$$

$$T = 2(t_{m1} - t_{p1})$$

$$w_n = \sqrt{\left(\frac{2\pi}{T}\right)^2 + \phi^2}$$

$$\xi = \frac{\phi}{w_n} \quad o \quad \sqrt{\frac{ln^2\varphi}{4\pi^2 + ln^2\phi}}$$

$$t_m = t_{p1} - \frac{T}{2}$$

1.3 MÈTODOS DE LAZO CERRADO

- **Métodos de identificación para sistemas sobre amortiguados basados en la información ultima**

Para la identificación de plantas por este método es necesario determinar los valores de ganancia última y periodo de oscilación ultima teniendo el sistema en

lazo cerrado. Esto quiere decir que se lleva la planta al límite de su estabilidad, capturando el valor del controlador del proporcional (K_u) que lo lleve a ese estado y el periodo de oscilación del sistema (T_u).

- **Método de Chen**

Este método aproxima el proceso mediante un modelo de primer orden más tiempo muerto. La ganancia del modelo de identificación está dada por:

$$K_p = \frac{\Delta y}{K_c(\Delta u - \Delta y)}$$

Donde Δy es el cambio del valor en estado estable que presenta la respuesta del sistema a una entrada tipo escalón, Δu es el cambio en el escalón de entrada y K_c es la ganancia del controlador para la prueba de lazo cerrado.

$$\tau = \frac{T_u}{2\pi}\sqrt{K_u^2 K_p^2 - 1}$$

$$t_m = \frac{T_u}{2\pi}\left[\pi - tan^{-1}\left(2\pi\tau/T_u\right)\right]$$

- **Método de Ho**

Este método identifica el proceso mediante un modelo de polo doble más tiempo muerto, las variables K_u, K_p, T_u se calculan igual que el método Chen.

$$\tau = \frac{T_u}{2\pi}\sqrt{K_u K_p - 1}$$

$$t_m = \frac{T_u}{2\pi\left[\pi - 2tan^{-1}\left(2\pi\tau/T_u\right)\right]}$$

- **Métodos de control P**

Los métodos basados en la información última tienen el inconveniente de que se lleva el sistema al límite de su estabilidad, por lo que se podría presentar una operación inestable en presencia de perturbaciones externas.

Este método se realizo con el objetivo de tener que realizar una prueba de lazo cerrado, estos métodos identifican un modelo para el proceso a partir de la respuesta del sistema ante un cambio en el valor deseado, cuando éste es controlado por un controlador proporcional, sin necesidad de llevarlo al límite de su estabilidad.

- **Método de Yuwana y Seborg (Vea Figura 63)**

Para poder aplicar este método la respuesta del sistema cuando se le aplica una entrada tipo escalón debe tener un comportamiento sub-amortiguado. Esta se modela como la respuesta de un sistema de 2 orden más tiempo muerto, y se identifica para el proceso un modelo de primer orden más tiempo muerto.

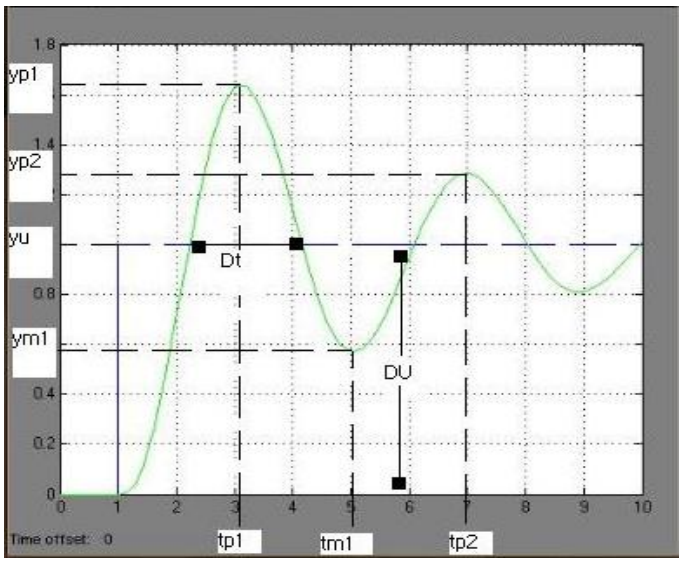


Figura 63. Metodo de Yuwana y Serborg

$$K_p = \frac{\Delta y}{K_c(\Delta u - \Delta y)}$$

Donde K_c es el valor del proporcional cuando la respuesta del sistema se comporta como sub-amortiguado.

$$k = K_c K_p$$

$$\tau = \frac{\Delta t}{\pi\left[\varsigma\sqrt{k+1} + \sqrt{\varsigma^2(K+1)+k}\right]\sqrt{(1-\varsigma^2)(k+1)}}$$

$$t_m = \frac{2\Delta t\sqrt{(1-\varsigma^2)(k+1)}}{\pi\left[\varsigma\sqrt{k+1} + \sqrt{\varsigma^2(k+1)+k}\right]}$$

Donde

$$\varsigma = \frac{[(\varsigma]_1 + \varsigma_2)}{2}$$

$$\varsigma_1 = \frac{-\ln\left[\frac{Y_u - Y_{m1}}{Y_{p1} - Y_u}\right]}{\sqrt{\pi^2 + ln^2\left[\frac{Y_u - Y_{m1}}{Y_{p1} - Y_u}\right]}}$$

$$\varsigma_2 = \frac{-\ln\left[\frac{Y_{p2} - Y_u}{Y_{p1} - Y_u}\right]}{\sqrt{4\pi^2 + ln^2\left[\frac{Y_{p2} - Y_u}{Y_{p1} - Y_u}\right]}}$$

ANEXO 4

1. GENERALIDADES DE MATLAB

El calificativo de MATLAB® procede de la contracción de los términos MATrix LABoratory y fue creado para el fácil acceso a las librerías que son de gran importancia en el área de la computación y el cálculo matricial.

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado, orientado para el desarrollo de proyectos con elevados cálculos matemáticos y la visualización gráfica de estos. MATLAB integra análisis numérico, cálculo matricial, procesado de señal, todo ello en un entorno fácil para el usuario.

Tanto en el ambiente académico como el industrial, Matlab se ha convertido en una herramienta básica para la resolución de complejos problemas matemáticos en diferentes áreas como la computación, el cálculo numero, prototipos de algoritmos, teoría de control automático, estadística etc.

Matlab consta de diferentes aplicaciones o toolboxes especializados orientados a ingenieros, científicos y todo tipo de profesionales técnicos. Entre ellos destacan: Sistemas de Control, Adquisición de Datos, Tiempo Real, Lógica Fuzzy, Procesamiento de Imagines, Redes Neuronales, Optimización, Procesamiento de Señal, etc.

El software Matlab dispone de dos herramientas adicionales que expanden sus prestaciones, a saber Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario – GUI).

1.1 HERRAMIENTA GUIDE

Para el desarrollo interactivo de la aplicación se cuenta con la herramienta de MATLAB llamada GUIDE (Graphical Use Interface Development Environment). Esta herramienta esta creada para desarrollar GUIs (Graphical User Interfaces) de manera fácil y rápidamente haciendo sencillo el diseño y presentación de los controles de la interfaz, reduciendo el trabajo en el momento de seleccionar, deshacer, arrastrar y centrar controles, así como la personalización de las propiedades de estos.

El asunto a seguir para el tratamiento de un programa mediante GUIDE es que una vez se tienen todas las funciones y controles en lugar, se editan las rutinas de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando sea necesario de utilizar. GUIDE está diseñado para realizar con menor dificultad el proceso de desarrollo de interfaz gráfica, para ello cuenta con un editor de propiedades (property editor) con el que podrá modificar en cualquier momento los nombres, valores por defecto y las propiedades de los elementos.
Para acceder a la herramienta GUIDE se tienen las siguientes maneras:

- Tecleando el comando >>guide desde el prompt de MATLAB

- A través del Launch Pad

- Llevando el puntero a File -> New -> GUI dentro del menú de MATLAB

Al arrancar la Aplicación GUIDE, al usuario le es presentada la pantalla de la Figura 64.

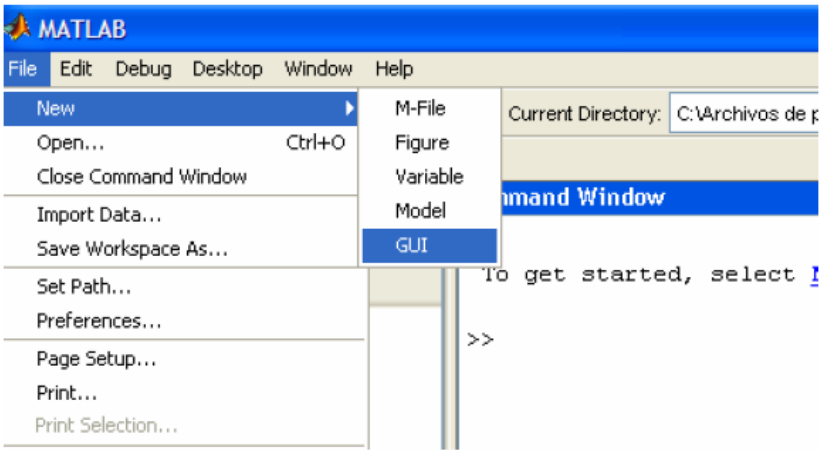


Figura 64. Inicio de la aplicación GUIDE

Una vez se inicie la aplicación aparece la siguiente interfaz, que se puede ver en la Figura 65, luego se asumirá si se desea realizar una nueva aplicación o bien abrir una existente. En el lugar que se cree una nueva aplicación se puede iniciar a partir de aplicaciones prediseñadas, como por ejemplo: con controles, con pantallas para gráficos y menú, o un cuadro de diálogo.

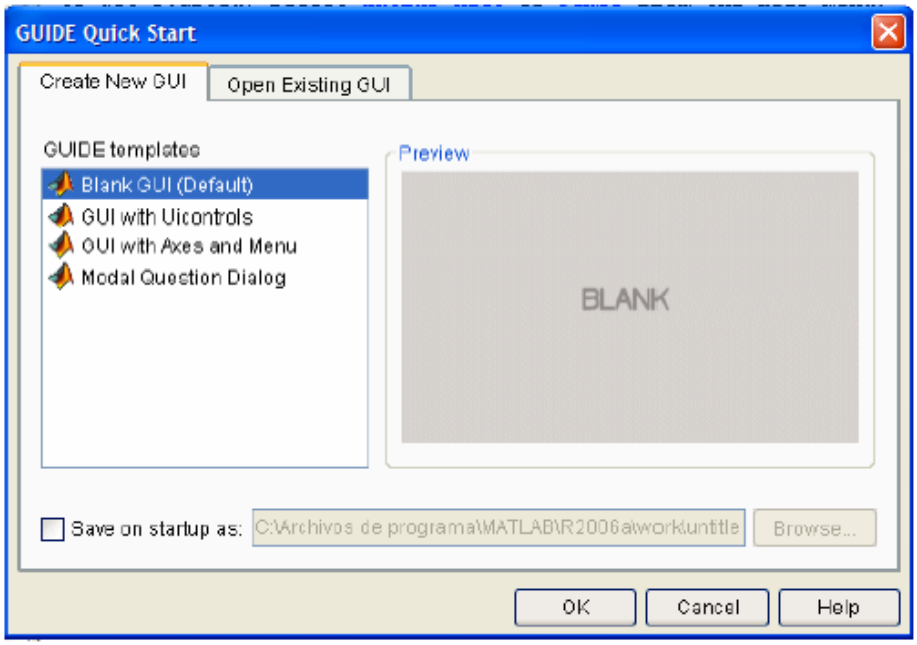


Figura 65. Menu de la herramienta GUIDE

Si se comienza con un diseño en blanco, se tiene la opción de acceder a los diferentes controles precisos para el desarrollo de la interfaz. Las correspondencias de los botones se han ordenado en la Tabla a modo de resumen con un breve extracto de su utilización. La Tabla 6 resume las herramientas de inicio de GUIDE

Al crear una GUI se generan dos ficheros:

- Un archivo con extensión .FIG, que es el que contiene los elementos gráficos así como las propiedades de la interfaz.
-
- Un archivo .M que es el contenedor del código con las esquelas de los botones de control de la interfaz. Cada vez que se introduzca un elemento gráfico en el .FIG se generará unas líneas de programa automáticamente asociadas a ese tipo de control. Estas líneas de programa son vacías, es decir, que requieren ser llenadas para llevar a cabo alguna acción durante su ejecución. Esta función es la callback mencionado anteriormente.

Item	Tag	Detalles
1	Select	Selección de un objeto
2	Push Button	Botón
3	Slider	Barra de desplazamiento
4	Radio Button	Botón de selección circular
5	Check Box	Botón de selección tipo lista
6	Edit Box	Casilla de edición
7	Static Text	Casilla de texto fijo
8	Pop-up menu	Menú desplegable
9	List Box	Menú en lista
10	Toggle Button	Botón con memoria
11	Axes	Pantalla para visualización de gráficos
12	Panel	Opción para generar un panel
13	Button Group	Opción para agrupar botones
14	Activex Control	Opción para generar controles ActiveX
15	Align objects	Opción para alinear objetos
16	Menu Editor	Editor de opciones de menú
17	Tab order Editor	Opción para cambiar el orden de las ventanas en el editor
18	M-file Editor	Editor de los ficheros .m
19	Property inspector	Editor propiedades de los controles
20	Object Browser	Buscador de Objetos
21	Run	Botón de compilación

Tabla 6. Resumen de las herramientas de inicio de GUIDE

1.2 IDENTIFICADORES (HANDLES)

Cada uno de los objetos de MATLAB tiene un identificador único a los que les llamara andel o id. Algunos gráficos tienen muchos objetos, en cuyo caso tienen múltiples handles, el objeto raíz (pantalla) es siempre único y su identificador siempre es cero. El identificador de las ventanas siempre es un entero que aparece en la barra de nombre de dicha ventana. Los identificadores de otros elementos son números tipo float.

En MATLAB puede haber variedad de ventanas abiertas pero sólo una está activa. Cada una de estas ventanas puede tener ejes abiertos, pero sólo se dibuja en los ejes activos. Los identificadores de la ventana activa, de los ejes activos y del objeto activo se pueden obtener con los siguientes comandos:

- gcf (get current figure): devuelve el entero que es el handle de la ventana activa.

- gca (get current axis): devuelve handle de los ejes activos.

- gco (get current object): devuelve handle del objeto activo
  .

delete handle: borra el objeto correspondiente y todas sus cualidades.

ANEXO 5

1. CODIGO DE BOTONES EN LA PLATAFORMA

1.1 PUSH BUTTON (RESPUESTA A UN STEP)

Figura 66. Boton de Respuesta a un Step

```
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global mode2
global cc
global xy
global xm
global dd
DIO=digitalio('nidaq','Dev1');
línea=addline(DIO,0,'out');
ai = analoginput('nidaq','Dev1');
addchannel(ai,0);
set(ai,'inputType','singleEnded')
set(ai,'samplerate',333);% 2.4segundos se estabiliza
n=10;
putvalue(DIO,0)
pause(n)
putvalue(DIO,1)
start(ai)
mode2=getdata(ai);
b=ones(1,10)/10;
yt1=filtfilt(b,1,mode2);
 cc=smooth(yt1,0.3);
 fu=length(cc);
```

```
dd=cc(950)
t1=0:3/999:3;
t=0:10/1499:10;
axes(handles.axes5)
plot(t1,cc)
zoom on
Grid
 %vamos a buscar el tiempo de establecimiento
 wwx=dd*0.63
% wwx=4.345*0.63;
for i=1:1:length(cc)
    if cc(i)<wwx
       m=cc(i)
      else
    end
end
 for ii=1:1:length(cc)
    if cc(ii)==m
        xm2=ii
    else
    end
end
 e=xm2+1
n=cc(e)
m5=m
rt1=t1(xm2)% punto1(rt1,m5)
rt2=t1(e)%  punto2(rt2,n)
m1=(n-m5)/(rt2-rt1)
b=m5-(m1*rt1)
xy=(wwx-b)/m1
set(handles.text6,'string',xy);
TS2=xy*4
set(handles.text7,'string',TS2);
 for i2=1:1:length(t)    %cambio el er, dependiendo del tamaño del vector
    if t(i2)<TS2  %cambio el ..por el valor del q deseo buscar
       xm=i2;
   else
```

```
    End
End
 delete(DIO)
clear DIO
delete(ai)
clear ai
```

1.2 LISTBOX (IDENTIFICACION EN LAZO ABIERTO)

Figura 67. Listbox para la identificacion en lazo abierto

```
% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ct1
global ct2
```

```
global a
global b
 c=gco;
    d=get(c)
    fun4 =get(c,'Value');
    if fun4==1
        a=0.91;
        b=1.26;
        ct1=0.25;
        ct2=0.75;
        elseif fun4==2
        a=0.55;
        b=2.80;
        ct1=0.28;
        ct2=0.40;
       elseif fun4==3
        a=0.67;
        b=1.29;
        ct1=0.35;
        ct2=0.85;
       elseif fun4==4
        a=0.14;
        b=1.54;
        ct1=0.33;
        ct2=0.67;
        elseif fun4==5
        a=0.15;
        b=1.50;
        ct1=0.28;
        ct2=0.63;
        elseif fun4==6
        a=1.24;
        b=1.49;
        ct1=0.33;
        ct2=0.7;
          end
% Hints: contents = get(hObject,'String') returns listbox2 contents as cell array
```

```
%       contents{get(hObject,'Value')} returns selected item from listbox2
```

1.3 PUSH BUTTON (IDENTIFICAR)

Figura 68. Boton para identificacion

```
% --- Executes on button press in pushbutton18.
function pushbutton18_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ct1
global ct2
global a
global b
global t
global kp
global TS2% esto viene
global T
global tm
global cc
global dd
global xy
deltau=5;
yu=dd;%%esto viene
kp=yu/deltau;
t1m=yu*ct1
t2m=yu*ct2
t=0:3/999:3;
 for i=1:1:length(cc)
    if cc(i)<t1m
     m(i)=cc(i)
     Else
```

```
    End
End
 for t11=1:1:length(m)
    if m(length(m))>m(length(m)-1)
    m5=m(length(m));
    End
end
e=t11+1
n=cc(e)
 rt1=t(t11)% punto1(rt1,m5)
rt2=t(e)%  punto2(rt2,n)
 m1=(n-m5)/(rt2-rt1)
b1=m5-(m1*rt1)
xy2=(t1m-b1)/m1
 %%%
for i1=1:1:length(cc)
    if cc(i1)<t2m
       mm(i1)=cc(i1);
    else
    end
end
 for t12=1:1:length(mm)
    if mm(length(mm))>mm(length(mm)-1)
    m55=mm(length(mm));
 end
end
e1=t12+1
n1=cc(e1)
 rt11=t(t12)% punto1(rt11,m55)
rt22=t(e1)%  punto2(rt22,n1)
 m11=(n1-m55)/(rt22-rt11)
b2=m55-(m11*rt11)
xy3=(t2m-b2)/m11
%%%%
%
 T=a*(xy3-xy2)
 polo=1/T
```

```
TS=4/polo
tm=b*xy2+(1-b)*xy3
TS2=xy*4;
TSG=TS2-TS %aki se mira si esta bien o no el polo
```

1.4 PUSH BUTTON (APLICAR A LA ENTRADA UN STEP)

Figura 69. Boton para aplicar a la entrada un Step

```
% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global T
global tm
global kp
global un
global xde
global x1
num=kp;
den=[T 1];
[num1,den1]=pade(tm,1);
x1=tf(num,den)
x2=tf(num1,den1);
x3=x1*x2
nu=num*num1;
 xde=conv([den],[den1]);
axes(handles.axes4)
step(x3)
Grid
zoom on
```

1.5 LISTBOX (FRECUENCIAS UTILIZADAS)

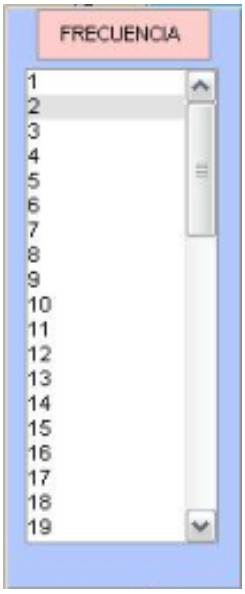


Figura 70. Listbox de las frecuencias utilizadas en la plataforma

```
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global r
global a
 c=gco;
    d=get(c)
    fun4 =get(c,'Value');
    if fun4==1
        r=0.8;
        a=1;
   elseif fun4==2
        r=0.82;
        a=2;
        elseif fun4==3
        r=0.84;
        a=3;
```

```
elseif fun4==4
    r=0.86;
    a=4;
  elseif fun4==5
    r=0.88;
    a=5;
   elseif fun4==6
    r=0.9;
    a=6;
   elseif fun4==7
    r=0.92;
    a=7;
 elseif fun4==8
  r=0.94;
    a=8;
  elseif fun4==9
    r=0.96;
    a=9;
    elseif fun4==10
    r=0.98;
    a=10;
  elseif fun4==11
    r=1;
    a=11;
  elseif fun4==12
    r=1.2;
    a=12;
    elseif fun4==13
    r=1.4;
    a=13;

elseif fun4==14
    r=1.6;
    a=14;
   elseif fun4==15
    r=1.8;
```

```
        a=15;
        elseif fun4==16
        r=2;
        a=16;
       elseif fun4==17
        r=2.2;
        a=17;
     elseif fun4==18
        r=2.4;
        a=18;
     elseif fun4==19
        r=2.6;
        a=19;
    elseif fun4==20
        r=2.8;%problem
        a=20;
      elseif fun4==21
        r=3;
        a=21;
     elseif fun4==22
        r=3.2;
        a=22;
   elseif fun4==23
        r=3.4;
        a=23;

    elseif fun4==24
        r=3.6;
        a=24;
    elseif fun4==25
        r=3.8;
        a=25;
    elseif fun4==26
        r=4;
        a=26;
  elseif fun4==27
        r=4.2;
```

```
        a=27;
    elseif fun4==28
        r=4.4;
        a=28;
   elseif fun4==29
        r=4.6;
        a=29;
   elseif fun4==30
        r=4.8;
        a=30;
  elseif fun4==31
        r=5;
        a=31;
   elseif fun4==32
        r=5.5;
        a=32;
   elseif fun4==33
        r=6;
        a=32;

        elseif fun4==34
        r=6.5;
        a=32;
       elseif fun4==35
        r=7;
        a=32;
        elseif fun4==36
        r=8;
        a=32;
        elseif fun4==37
        r=9;
        a=32;
        elseif fun4==38
        r=10;
        a=32;
        elseif fun4==39
        r=15;
```

```
    a=32;
    elseif fun4==40
    r=20;
    a=32;
    elseif fun4==41
    r=25;
    a=32;
    elseif fun4==42
    r=30;
    a=32;
   elseif fun4==43
    r=35;
    a=32;

    elseif fun4==44
    r=40;
    a=32;
   elseif fun4==45
    r=50;
    a=32;
   elseif fun4==46
    r=60;
    a=32;
   elseif fun4==47
    r=70;
    a=32;
  elseif fun4==48
    r=80;
    a=32;
  elseif fun4==49
    r=90;
    a=32;
    elseif fun4==50
    r=100;
    a=32;
End
 % Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
```

```
%       contents{get(hObject,'Value')} returns selected item from listbox1
```

1.6 PUSH BUTTON (LSIM, SIMULACION)

Figura 71. Boton para obtener la simulacion del sistema identificado a entradas sinusoidales

```
% --- Executes on button press in pushbutton20.
function pushbutton20_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global un
global xde
global r
global a
global xm
global xg
global xf2
global t233
global y
rr=10/r;
frec(a)=1/(rr)
ad=frec(a);
t=(0:10/1499:10);
u=2.5*sin(2*pi*ad*t);
[y,x]=lsim(nu,xde,u,t);
```

```
t2=t(xm:1500);
y2=y(xm:1500);
u2=u(xm:1500);
axes(handles.axes3)
plot(t2,u2)
zoom on
Grid
axes(handles.axes4)
plot(t2,y2)
zoom on
grid t233=y2;
```

1.7 PUSH BUTTON (SALIDA, ENVIO DE SEÑAL AL MOTOR)

Figura 72. Boton para el envio de señal al motor

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global mode
global t
global m7
global r
global a
global datos
global frec
global x
global b
global pasos
global rr
```

```
global xm
global ac
global xe
global tt2
ao = analogoutput('nidaq','Dev1');
addchannel(ao,0);
ai = analoginput('nidaq','Dev1');
addchannel(ai,0);
 set(ai,'inputType','singleEnded')
set(ai,'samplerate',300);
 d=10;
set(ao,'samplerate',150)
actualrate=get(ao,'samplerate');
lens=actualrate*d;
pasos=lens-1;
datos =2.5*sin(linspace(0,2*pi*r,lens))+2.5;
t=(0:d/pasos:d)
for i=1:length(datos)
            putsample(ao,datos(i))
            mode(i)=getsample(ai);
End
 delete(ao)
clear ao
delete(ai)
clear ai

 tt2=t(xm:1500);% recorte de tiempo, desde t11 que es el ts
er=(1500-xm)+1;
t23=datos(xm:1500)%recorte de datos, desde t11 que es ts
axes(handles.axes1)
 plot(tt2,t23)
zoom on
Grid
 for i2=1:1:length(t23)     %cambio el er, dependiendo del tamaño del vector
    if t23(i2)<2.5   %cambio el 2.5..por el valor del q deseo buscar
       m(i2)=t23(i2);
      else
```

```
    end
end
  if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)>0
            m5=m(tt3); %abajo
            e=tt3-1;
            n=t23(e);%arriba
            re1=tt2(e)%  punto1(re1,n)
            re2=tt2(tt3)% punto2(re2,m5)
            Break
        Else
          End
    End
Else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1)
            e=tt3;
            n=t23(e);%arriba
            re1=tt2(e-1)% punto1(re1,m5)
            re2=tt2(tt3)%  punto2(re2,n)
            Break
        Else


        End
    End
End
 m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x=(2.5-b)/m1
 rr=10/r;
frec(a)=1/(rr)
ac=frec(a);
m7(a)=x
 xc=max(datos);
xd=min(datos);
```

```
xe=(xc-xd)/2;
set(handles.text2,'string',xe);
```

1.8 PUSH BUTTON (ENTRADA, ADQUISICION DE LA SEÑAL GENERADA POR EL MOTOR)

Figura 73. Boton para la adquisicion de la señal generada por el motor

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global mode
global a
global t
global A
global m5
global xe
global xm
global t68
global xg
global xf2
cc=smooth(mode,'moving');
 tt2=t(xm:1500);% recorte de tiempo, desde t11 que es el ts
er=(1500-xm)+1;
t23=cc(xm:1500);%recorte de datos, desde t11 que es ts..yt va a ser cc
 t68=cc-median(cc);
 for i2=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
    if t68(i2)<0   %cambio el 2.5..por el valor del q deseo buscar
       m(i2)=t68(i2);
    Else
    End
End
```

```
 if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)<0
            m5=m(tt3); %abajo
            e=tt3;
            n=t68(e);%arriba
            re1=tt2(e);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)

         Break
        Else
         End
    End
Else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1);
            e=tt3-1;
            n=t68(e);%arriba
            re1=tt2(e-1);% punto1(re1,m5)
            re2=tt2(tt3);%  punto2(re2,n)
            Break
        Else
        End
    End
end
 if m(1)==0
mm=m(tt3:length(m));
for ttt3=1:1:length(mm)
    if mm(ttt3)<0
        m55=mm(ttt3);
        e1=(ttt3-1)+tt3;
%           nn=y(e1)%arriba
%           re11=t2(e1)%  punto1(re1,n)
           re22=tt2(ttt3);% punto2(re2,m5)
           ty2=tt3;
        pn=m(tt3:e1);
```

```
    Else
        Break
    End
End
Else
    mm2=m(1:e)
    pt=mm2(1);
    ty2=1;
    pn=m(1:e);
End
 picopordebajo=min(pn)%amplitud de la señal
n4=picopordebajo*-1
 for ty=1:1:length(pn)
    if pn(ty)==picopordebajo
        ty1=ty;
    Else
    End
End
 tye=ty1-1;
tye1=ty2+tye;
tiem=tt2(tye1);%tiempo de la amplitud de la señal
 %%%pico por encima
 for i22=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
    if t68(i22)>0   %cambio el 2.5..por el valor del q deseo buscar
       m(i22)=t68(i22);
     Else
    End
End
 if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)>0
            m5=m(tt3); %abajo
            e=tt3;
            re1=tt2(tt3-1);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)
            Break
        Else
```

```
        End
    End
Else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1);
            e=tt3-1;
            re1=tt2(tt3-1)% punto1(re1,m5)
            re2=tt2(tt3)%  punto2(re2,n)
            break
        else
         end
    end
end
 if m(1)==0
mm=m(tt3:length(m))
for ttt3=1:1:length(mm)
    if mm(ttt3)>0
        m55=mm(ttt3);
        e1=ttt3-1;
%            nn=y(e1)%arriba
            %re11=t2(e1-1)%  punto1(re1,n)
            %re22=t2(ttt3)% punto2(re2,m5)
            e11=tt3+e1;
            pn=m(tt3:e11);
            ty2=tt3;
    else
        break
    end
end
else
    mm2=m(1:length(m));
pt=mm2(1);
pn=m(1:e);
ty2=1;
End
 picoporencima=max(pn)%amplitud de la señal es esto - n3(popu), eso me da aprox 0.2372
```

```
xgg2=picoporencima-picopordebajo
xgg=(xgg2)/2;
axes(handles.axes2)
plot(tt2,xgg)
zoom on
Grid
 A(a)=20*log10(xgg/xe)
```

1.9 BOTONES PARA EL ANALISIS DEL ESPECTRO

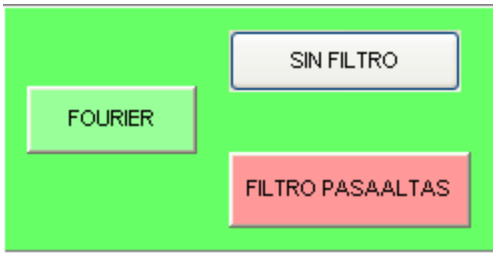


Figura 74. Botones para el analisis del espectro

1.9.1 PUSH BUTTON (FOURIER)

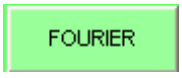


Figura 75. Boton para obtener el espectro de frecuencia de la señal trabajada

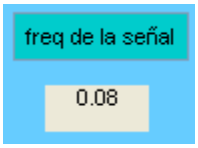


Figura 76. Frecuencia trabajada (0.08 Hz)

```
% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
global t68
global ac
global a
global frecc
L=length(t68);
Fs=150;
 NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(t68,NFFT)/L;
fp = Fs/2*linspace(0,1,NFFT/2+1);
 % Plot single-sided amplitude spectrum.
aa=abs(Y(1:NFFT/2+1));
axes(handles.axes6)
plot(fp,2*aa)
zoom on
 Grid
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
xc=max(2*aa)
xd=2*aa;
 for i2=1:1:length(xd)
    if xd(i2)==xc   %cambio el ..por el valor del q deseo buscar
       m=xd(i2)
 Else
    End
End

frecuenciadelasenal=xc
frecc(a)=xc
set(handles.text8,'string',frecuenciadelasenal);
```

1.9.2 PUSH BUTTON (SIN FILTRO)

Figura 77. Boton para obtener la señal sin utilizar el filtro

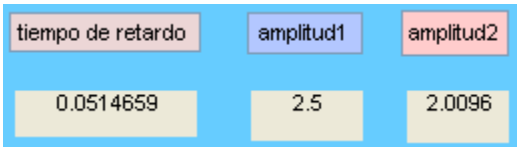


Figura 78. Variables a obtener con el boton Sin Filtro

Fuente: Diseño del autor

```
% --- Executes on button press in pushbutton23.
function pushbutton23_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a
global xg
global amplitud
global t233%señal
global x
global tt2
global ac
global we
global wx
global w
for i2=1:1:length(t233)     %cambio el er, dependiendo del tamaño del vector
   if t233(i2)<0   %cambio el 2.5..por el valor del q deseo buscar
      m(i2)=t233(i2);
Else
       m(i2)=0;
   End
End
 if m(1)==0
```

```
    for tt3=1:1:length(m)
        if m(tt3)<0
            m5=m(tt3) %abajo
            e=tt3-1;
            n=t233(e);%arriba
            re1=tt2(e);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)

            break
        else
      end
    end
else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1)
            e=tt3;
            n=t233(e);%arriba
            re1=tt2(e-1);% punto1(re1,m5)
            re2=tt2(tt3);%  punto2(re2,n)
            break
        else
     end
    end
 end
%%%%%%%%%%%%%%%%%%%%%%%%
m2=m(tt3:length(m));
ty2=tt2(tt3:length(m));
if a>11
 if m2(1)==0
    for ttr3=1:1:length(m2)
        if m2(ttr3)<0
            m56=m2(ttr3) %abajo
            e6=ttr3-1;
            tty1=ty2(ttr3)
            tty2=ty2(e6)
            Break
```

```
        Else

      End
    End
Else
    for ttr3=1:1:length(m2)
        if m2(ttr3)==0
            m56=m2(ttr3-1)
            e6=ttr3;
            tty1=ty2(ttr3-1)
            tty2=ty2(e6)
            Break
        Else
  End
    End
 End
for i24=1:1:length(tt2)
    if tt2(i24)==tty1
        mt1=i24;
         break
    Else
    End
end
for i25=1:1:length(tt2)
    if tt2(i25)==tty2
        mt2=i25;
         break
    Else
    End
end
m561=t233(i24);
m562=t233(i25);
 m11=(m562-m561)/(tty2-tty1);
b=m561-(m11*tty1);
x2=(0-b)/m11
m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
```

```
x3=(0-b)/m1
if x3>x
   x4=x3-x;
Else
   x4=x2-x;
End
Else
m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x3=(0-b)/m1
x4=x3-x;
end
 set(handles.text1,'string',x4);
 amplitud(a)=xg
set(handles.text3,'string',xg);
w(a)=2*pi*ac
we(a)=-ac*x4
wr=we(a)
wx(a)=(wr*180)/pi
```

1.9.3 PUSH BUTTON (FILTRO PASA ALTAS)

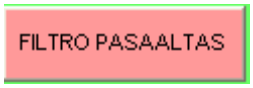


Figura 79. Boton para aplicar un filtro pasa altas

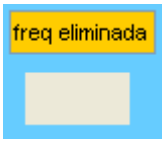


Figura 80. Visualizacion de frecuencia eliminada con el filtro pasa alta

```
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global tt2
global cc
global xg
global xf2
global a
global amplitud
global xe
global t68
global A
global ac
global wx
global y2
global we
global t233
ad=ac-0.15;
fn=ad/ac;
fc=fn/(ac/2);
[b,c]=butter(1,fc,'high');
yt=filtfilt(b,c,cc);
cc3=smooth(yt);
 cc2=smooth(cc3,'moving');
cc4=smooth(cc2,0.01);
t26=cc4(xm:1500);
 t68=cc4-median(cc4);
 for i2=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
    if t68(i2)<0   %cambio el 2.5..por el valor del q deseo buscar
       m(i2)=t68(i2);
  Else
    End
End
 %pico por debajo
 if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)<0
            m5=m(tt3); %abajo
```

```
            e=tt3;
            n=t68(e);%arriba
            re1=tt2(e);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)
         Break
        Else
       End
    End
Else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1)
            e=tt3-1;
            n=t68(e);%arriba
%               re1=tt2(e-1);% punto1(re1,m5)
%               re2=tt2(tt3);%  punto2(re2,n)
            Break
        Else
  End
    End
End
if m(1)==0
mm=m(tt3:length(m));
for ttt3=1:1:length(mm)
    if mm(ttt3)<0
        m55=mm(ttt3);
        e1=(ttt3-1)+tt3;
%               nn=y(e1)%arriba
%               re11=t2(e1)%  punto1(re1,n)
%               re22=tt2(ttt3);% punto2(re2,m5)
            ty2=tt3;
        pn=m(tt3:e1);
    Else
        Break
    End
End
Else
```

```
    mm2=m(1:e)
    pt=mm2(1);
    ty2=1;
pn=m(1:e);
End

picopordebajo=min(pn)%amplitud de la señal
n4=picopordebajo*-1
for ty=1:1:length(pn)
    if pn(ty)==picopordebajo
        ty1=ty;
    Else
    End
End
 tye=ty1-1;
tye1=ty2+tye;
tiem=tt2(tye1);%tiempo de la amplitud de la señal
 %%%pico por encima
 for i22=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
    if t68(i22)>0   %cambio el 2.5..por el valor del q deseo buscar
       m(i22)=t68(i22);
    Else
    End
End
 if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)>0
            m5=m(tt3); %abajo
            e=tt3;
            re1=tt2(tt3-1)%  punto1(re1,n)
            re2=tt2(tt3)% punto2(re2,m5)
      Break
        Else
end
    end
else
    for tt3=1:1:length(m)
```

```
        if m(tt3)==0
            m5=m(tt3-1)
            e=tt3-1;
            re1=tt2(tt3-1)% punto1(re1,m5)
            re2=tt2(tt3)%  punto2(re2,n)
            Break
        Else
     End
    End
End
if m(1)==0
mm=m(tt3:length(m))
for ttt3=1:1:length(mm)
    if mm(ttt3)>0
        m55=mm(ttt3)
        e1=ttt3-1;
%            nn=y(e1)%arriba
            %re11=t2(e1-1)%  punto1(re1,n)
            %re22=t2(ttt3)% punto2(re2,m5)
            e11=tt3+e1;
            pn=m(tt3:e11);
            ty2=tt3;
    Else
        Break
    End
End
Else
    mm2=m(1:length(m));
pt=mm2(1);
pn=m(1:e);
ty2=1;
End
picoporencima=max(pn)%amplitud de la señal es esto - n3(popu), eso me da aprox 0.2372
xgg2=picoporencima-picopordebajo
xgg=(xgg2)/2;
%%%%%%%%
for i2=1:1:length(t233)     %cambio el er, dependiendo del tamaño del vector
```

```
    if t233(i2)<2.5   %cambio el 2.5..por el valor del q deseo buscar
       m(i2)=t233(i2);
Else
    End
End
  if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)>0
            m5=m(tt3); %abajo
            e=tt3-1;
            n=t233(e);%arriba
            re1=tt2(e);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)
         Break
        Else
      End
    End
Else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1);
            e=tt3;
            n=t233(e);%arriba
            re1=tt2(e-1);% punto1(re1,m5)
            re2=tt2(tt3);%  punto2(re2,n)
            Break
        Else
   End
    End
End
 m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x2=(2.5-b)/m1
x3=x2-x;
 %%%%%%%%%%
axes(handles.axes2)
plot(tt2,y2)%%%ojo se cambio
```

```
zoom on
Grid
amplitud(a)=xg
set(handles.text3,'string',xg);
we(a)=-ac*x3;
wr=we(a);
wx(a)=wr*180/pi
A(a)=20*log10(xg/xe)
```

1.10 PUSH BUTTON (COMPARACION DE BODE)

Figura 81. Visualizacion del Diagrama de Bode

```
% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global w
global A
global un
global xde
global wx
Figure
semilogx(w,A)
Grid
zoom on
Figure
semilogx(w,wx)
Grid
zoom on
Figure
bode(nu,xde)
Grid
zoom on
```

ANEXO 6

1. COMANDOS Y PROPIEDADES UTILIZADAS:

1.1 COMANDOS

- **Getsample:**

Inmediatamente adquirir una muestra de la entrada analógica

Sintaxis

muestra = getsample (obj)

Alegaciones

Obj: un objeto de entrada analógica.

Muestra: un vector fila que contiene una muestra por cada canal de contenidos por obj.

Descripción

muestra = getsample (obj) inmediatamente devuelve un vector fila que contiene una muestra por cada canal contenida por obj.

Comentarios

Usando getsample es una buena manera de probar la configuración de entrada analógica. Además, Getsample devuelve una muestra por cada canal Y no es compatible con tarjetas de sonido

- **Putsample:**

Inmediatamente envía una muestra a la salida analógica

Sintaxis

putsample (obj, datos)

Alegaciones

Obj Un objeto de salida análoga

Datos los datos que se ponen en la cola en el motor

Descripción

putsample (obj, datos) inmediatamente envía los datos vector fila, que consiste en una muestra por cada canal de contenidos por obj.

Comentarios

Usando putsample es una buena manera de probar la configuración de salida analógica. Además:

- putsample no almacena las muestras en el motor de adquisición de datos.
- putsample puede ser ejecutado en cualquier momento a partir de los canales se han añadido a obj.
- putsample no es compatible con tarjetas de sonido

- **Getdata:**

Extraer datos de entrada analógica, antes de adquirir datos con el getdata, primero se debe inicializar el puerto de entrada análoga

```
start(ai)
mode2=getdata(ai);
```

- **Filtfilt:**

Este filtro digital reduce al mínimo la puesta en marcha y termina transitorios, haciendo coincidir las condiciones iniciales. Primero filtra el vector x, y su respuesta la rota y le vuelve a aplicar el mismo filtro. La respuesta final evita la distorsión de fase propia de los filtros IIR.

y = filtfilt (b, a, x)

El vector b proporciona los coeficientes de numerador del filtro y los coeficientes del denominador.

- **Pade:**

Este comando se utiliza para aproximar los retardos de un modelo, hay que tener en cuenta que la aproximación de Padé tiene ganancia unitaria en todas las frecuencias. Este comando se utiliza de la siguiente manera:

pade(T, N)

Donde T es el tiempo de retardo y N es el orden del sistema.

- **LSIM**

Con la función LSIM se puede simular la respuesta en el tiempo de un modelo a entradas arbitrarias. En este caso se ingresa ondas sinusoidales de diferentes frecuencias, generando otras ondas sinusoidales de igual frecuencia, pero de amplitud y fase diferente (Vea Figura 82). Para la demostración, se ingresa el numerador y el denominador de la función de transferencia de un sistema, se especifica el eje de tiempo, y la señal que se va a enviar a la entrada del sistema, en este caso es una sinusoidal.

```
w = 0.25
num = 0.8481;
den = [0.4781 1];
[num1,den1]=pade(0.012,1);
num2=num*num1;
den2=conv([den],[den1]);
t = 0:10/1499:10;
u = 2.5*sin(2*pi*w* t);
[y, x] = lsim (num2, den2, u, t);
```

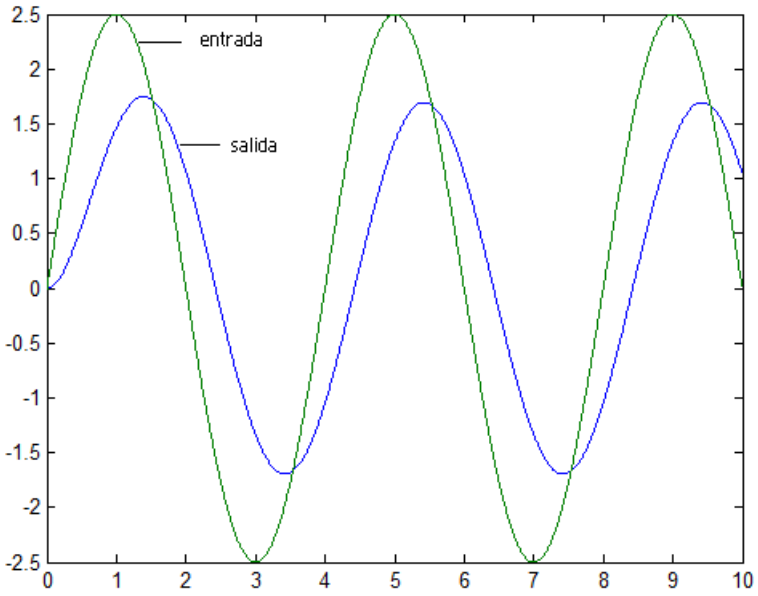


Figura 82. Señal simulada con LSIM

- **SMOOTH**

Es un filtro que tiene como propósito, el suavizar la señal reduciendo el ruido blanco al azar mientras se mantiene la respuesta al escalón más aguda, especificando el método de filtro de media móvil ('moving'). Este es configurado de la siguiente manera:

```
yy=smooth(y)
yy=smooth(y,span)
```

yy=smooth(y,*method*)

yy=smooth(y,span, *method*)

El span establece la duración de la media móvil a palmo. span debe ser impar.
Y los métodos son varios, entre ellos se encuentran:

- **Moving average (default).** Un filtro de paso bajo con coeficientes del filtro igual a la inversa de la duración.
- **Lowess:** regresión lineal ponderada local por mínimos cuadrados y un grado primero modelo polinomial.
- **loess**: regresión lineal ponderada local por mínimos cuadrados y un 2° grado modelo polinomial.
- **sgolay:** Un promedio móvil generalizada con coeficientes del filtro determinado por un lineal de regresión por mínimos cuadrados ponderados y un modelo polinomial de grado específico (por defecto es 2). El método puede aceptar datos no uniforme predictores.
- **rlowess:** Una sólida versión de 'LOWESS' que asigna menor peso a los valores atípicos en la regresión. El método asigna cero peso a los datos fuera de seis decir desviaciones absolutas.
- **Rloess:** Una sólida versión de "loess" que asigna menor peso a los valores atípicos en la regresión. El método asigna cero peso a los datos fuera de seis decir desviaciones absolutas.

- **FFT**

Se demuestra que cualquier proceso periódico se puede modelar, con la precisión deseada, mediante series de términos de funciones sinusoidales (seno y coseno), lo que se conoce como **series de Fourier**, y se denomina **espectro** a la representación de las amplitudes, en el eje de las Y, que constituyen los diferentes términos de la serie para toda la gama de frecuencias (eje de las X). En matlab a la fft se obtiene de la siguiente manera:

Y = fft (x)

Con esta función se devuelve transforma de Fourier discreta (DFT) de vectores X, calculado con una transformación rápida de Fourier (FFT) . Luego se saca la magnitud de la fft con abs, y se grafica frecuencia vs la magnitud de la fft. Para demostrarlo, se coge una señal donde se mezclan 2 señales sinusoidales de diferente frecuencia, una de 10hz y otra de 60 hz, y se observa que con el espectro de frecuencia se pueden visualizar las frecuencias. La Figura 83 muestra la señal con interferencia y la Figura 84 las magnitudes del espectro en frecuencia.[1][2][3]

```
Fs=150;%frecuencia de muestreo
t=0:10/(1500-1):10;
s1=sin(2*pi*60*t)+sin(2*pi*10*t);
plot(t,s1)
L=length(s1);

NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(s1,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
plot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
```

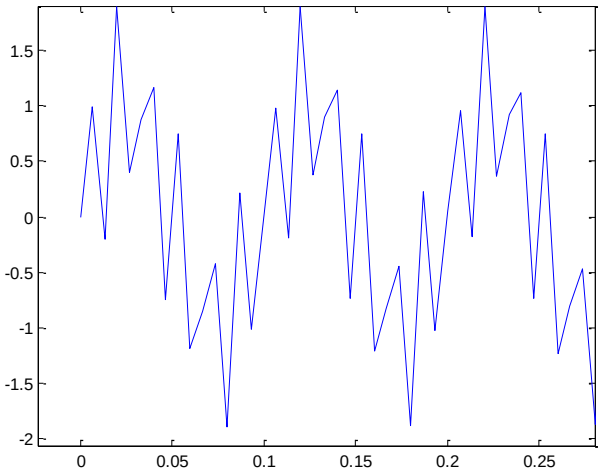


Figura 83. Señales sumadas

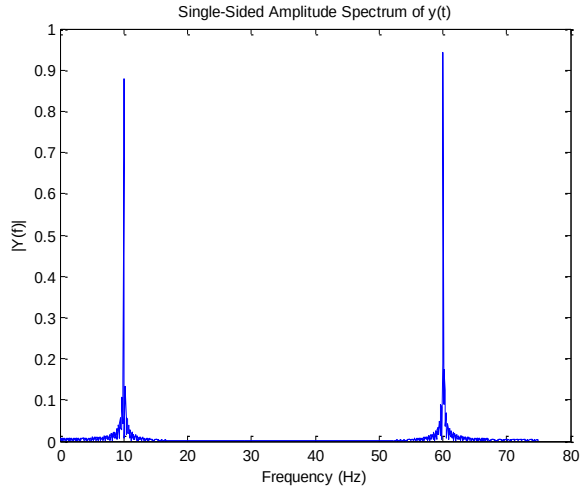


Figura 84. Espectro de frecuencias

- **FILTRO BUTTER**

Filtros de Butterworth se caracterizan por una respuesta de magnitud que es máximamente plana en la banda de paso y monótona en general. Estos filtros pueden ser utilizados como filtro pasa bajo, pasa alta y pasa banda, dependiendo del tipo que se escoja. Se configura de la siguiente manera.

(zp)= butter(n,Wn), diseña un paso bajo de orden n digital filtro Butterworth normalizado con frecuencia de corte Wn. Devuelve los coeficientes del filtro.

(zp) = butter(n,Wn, 'flype'), diseña un paso alto, paso bajo, o filtro de banda de detención, donde *ftype 'la cadena'* es 'high' , 'low' , or 'stop', como se describe anteriormente. Para el 'stop' se especifican las2 frecuencias de la siguiente manera remplazando Wn, [Wn1 Wn2].

En este caso se uso un filtro pasa alto debido a que se tiene que eliminar una baja frecuencia intrusa en la señal. Para poder eliminarla se debe tener cuidado al escoger el valor de Wn debido a que este valor es normalizado, este procedimiento se hace de la siguiente manera:

$$Wn = \frac{Wc}{\left(\frac{frecuencia\ de\ la\ señal}{2}\right)}$$

Donde $Wc = \frac{frecuencia\ a\ eliminar}{frecuencia\ de\ la\ señal}$

Ahora sólo es reemplazar Wn en el filtro.

Como ejemplo, se le aplicó fft a una señal, el cual generó el siguiente espectro de frecuencia.(Vea Figura 85).

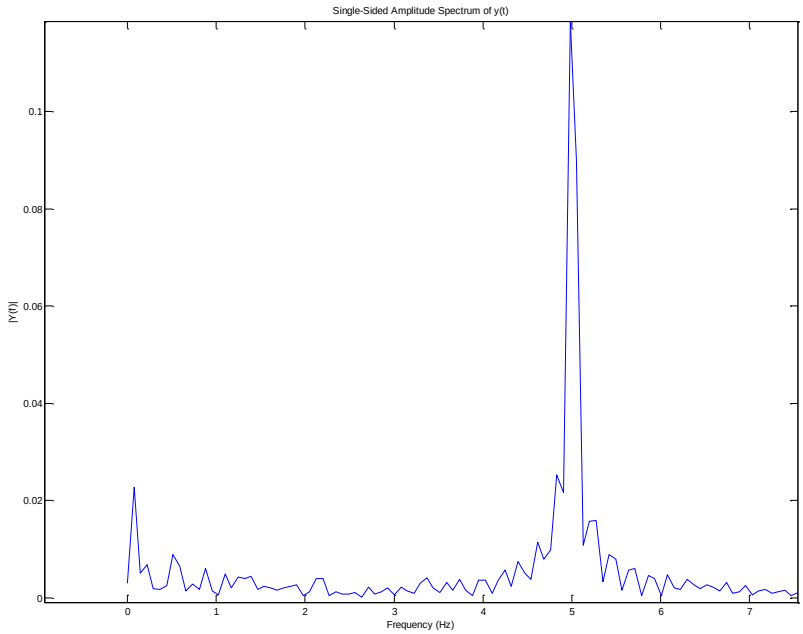


Figura 85. FFT aplicado a la señal

Se observa que la señal posee una frecuencia de 5hz, y ahí una pequeña frecuencia metida en la señal, la cual se desea eliminar. Después de eliminada se obtiene el espectro de la Figura 86.

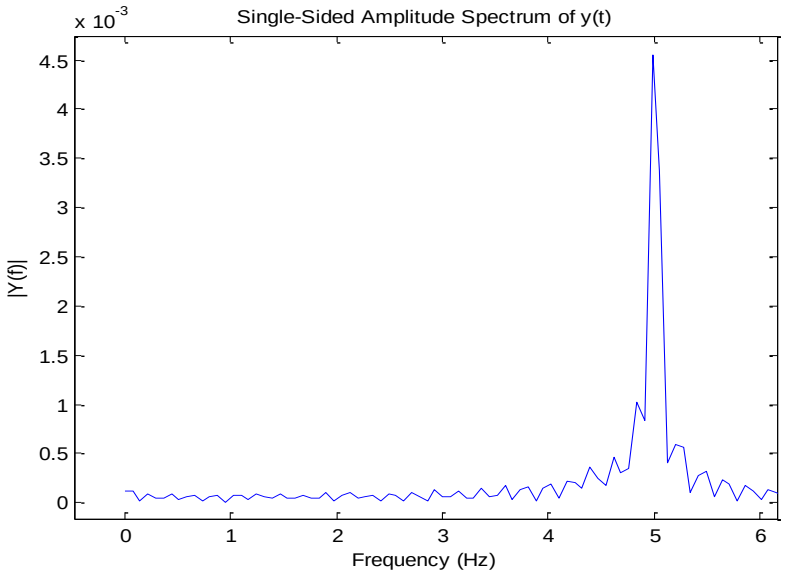


Figura 86. Espectro limpio de frecuencias bajas

- **INVFREQS**

Es útil en la conversión de la magnitud y los datos de fase en las funciones de transferencia. [b, a] = invfreqs (h, w, n, m) devuelve el numerador y el denominador en una función de transferencia cuya respuesta en frecuencia compleja figura en el vector h y w. Donde n es el orden del numerador y m el orden del denominador. Como ejemplo se tiene:

```
a = [1 2 3 2 1 4]; b = [1 2 3 2 3];
[h,w] = freqs(b,a,64);
[bb,aa] = invfreqs(h,w,4,5)
bb =
   1.0000   2.0000   3.0000   2.0000   3.0000
aa =
   1.0000   2.0000   3.0000   2.0000   1.0000   4.0000
```

Existen varias formas de leer y enviar señales desde matlab. Una muy conocida es por medio del puerto serial o paralelo. Debemos tener claro que el puerto paralelo se puede usar simultáneamente con dos o más programas a la vez, en cambio el puerto serial será de uso exclusivo de un solo programa. Si se va a utilizar el puerto serial, Para escribir datos usamos la función **fprintf** y para leer los datos del puerto se usa la función **fscanf**. En cambio si se va a utilizar el puerto paralelo para poder obtener el dato de entrada se coloca dato3=getvalue(dato2), y si se desea asignar un dato al puerto se coloca putvalue(dato,255). Otra forma de enviar y recibir señales es por medio del puerto USB con una aplicación enviando datos por medio inalámbrico a través de un modem hacia un pc u otro instrumento.

1.2 PROPIEDADES

- **Daqhwinfo:**

Es usada para examinar los recursos del hardware de adquisición de datos que son visibles para la Toolbox. Incluyendo las tarjetas instaladas, los controladores y adaptadores de hardware. La información retornada por éste comando depende de los argumentos suministrados.

Para mostrar la información general de la Toolbox digite:

```
Info=daqwinfo
```

Este comando muestra el nombre y la versión de la Toolbox, la versión que tiene instalada de MATLAB y el número de los adaptadores instalados; de esta forma:

```
Info =
ToolboxName:  'Data Acquisition Toolbox'
ToolboxVersion:   '2.10 (R2007a)'
MATLABVersion:   '7.4 (R2007a)'
InstalledAdaptors:   {3x1 cell}
```

Para mirar cuáles adaptadores están instalados se usa el objeto Info con el campo InstalledAdaptors de la forma:

```
Info.InstalledAdaptors
ans =
'nidaq'
'parallel'
'winsound'
```

- **SingleEnded:**

Por defecto el valor de la propiedad del tipo de entrada es diferencial (Differential), pero se puede establecer para nodo simple si así lo requiere la aplicación.
set(AI,'InputType', 'SingleEnded')

El valor de InputType determina el número de canales del hardware que pueden agregarse a un objeto. Si se toma el valor por defecto (Differential), el máximo número de canales que puede contener el objeto es 4, mientras que al establecer el valor de InputType para entradas de simple nodo (SimpleEnded) los 8 canales de entrada análoga que tiene la tarjeta pueden ser agregados al objeto.

- **Samplerate:**

Con la propiedad SampleRate se especifica la tasa de muestreo, teniendo en cuenta lo siguiente:

- La tasa de muestreo debe ser dos veces mayor al valor del componente de frecuencia máxima de la señal.

- La tasa máxima a los cuales los canales son muestreados depende el número de canales contenidos por el objeto, está dada por la fórmula,

$$Tasa\ \ máxima\ \ de\ \ Muestreo\ \ por\ \ Canal = \frac{Tasa\ \ máx\ \ de\ \ la\ \ Tarjeta}{Número\ \ de\ \ Canales}$$

- **Linspace:**

La función linspace genera vectores linealmente espaciadas. Es similar al operador dos puntos ":", pero ofrece un control directo sobre el número de puntos.

y = linspace (a, b,n) genera un vector fila y de n puntos linealmente espaciadas e incluyendo a y b.