UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Ingeniería Sistemas y Automática

IDENTIFICACIÓN DE SISTEMAS DINÁMICOS MEDIANTE TEST DE RELÉ MODIFICADO

TRABAJO FIN DE GRADO

AUTOR: TAMARA SERRANO SIMÓN

DIRECTOR: CONCEPCIÓN ALICIA MONJE MICHARET

Febrero, 2014

TRABAJO FIN DE GRADO

IDENTIFICACIÓN DE SISTEMAS DINÁMICOS MEDIANTE TEST DE RELÉ MODIFICADO

Por

Tamara Serrano Simón

Presentado en la

ESCUELA POLITÉCNICA SUPERIOR

de la

UNIVERSIDAD CARLOS III DE MADRID

Para la obtención del

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Directora de Trabajo Fin de Grado

Dña. Concepción Alicia Monje Micharet

Madrid, 22 Febrero 2014

TRABAJO FIN DE GRADO

IDENTIFICACIÓN DE SISTEMAS DINÁMICOS MEDIANTE TEST DE RELÉ MODIFICADO

Por

Tamara Serrano Simón

Directora de Trabajo Fin de Grado

Dra. Concepción Alicia Monje Micharet

TRIBUNAL CALIFICADOR

Presidente

Dr. Villa Briongos, Javier

Secretario

Dr. Sánchez Montero, David Ricardo

Vocal

Dr. Jardon Huete, Alberto

Madrid, 23 Febrero 2014

TRABAJO FIN DE GRADO

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTOS

Una pequeña reflexión hasta coger el sueño hizo que me inspirase en este apartado.

Muchas personas quizás consideren que es un apartado sin importancia donde con cuatro palabras bien dichas se rellena. Sin embargo, después de pensarlo mucho, considero que para mí resulta ser uno de los apartados más importantes de la memoria, ya que en la medida de lo posible me permite agradecer el esfuerzo a todas esas personas que han estado detrás durante todo este tiempo de trabajo. Puesto que seguramente nadie les felicite y valore, seré yo personalmente la que me encargue de hacerlo.

En primer lugar, como responsable y uno de los pilares de este trabajo debo nombrar a Dña. Concepción Alicia Monje Micharet, debo agradecerla todo la ayuda y disponibilidad que me ha ofrecido todo este tiempo, al igual que el apoyo que me ha dado en los momentos de desesperación. Desde el corazón y sin ser pelota considero que es una grandísima persona tanto personal como profesionalmente, no puedo evitar decir que admiro su inteligencia.

En segundo lugar mencionaré a mi familia la cual ha sufrido desde el primer día todo el trabajo. Agradeceré a mi padre y mi hermana todo su esfuerzo por intentar ayudarme y aguantarme en mis días insoportables. Pero sin duda a la que más debo agradecer y la nombraré en letras mayúsculas es a mi MADRE, ella ha aguantado todos mis cabreos, mis momentos de desesperación, mi falta de tiempo para ayudarla,… sin más ella ha sido mi otro pilar del trabajo y tengo tanto que agradecerla que las palabras en casa se me quedan cortas. Sin su apoyo y sus ánimos igual no hubiese llegado a terminar el trabajo.

Y para terminar también agradecerle a mi compañero y amigo Pedro que desde el primer día ha confiado en mí, me ha ayudado y me ha dado muchos ánimos.

TAMARA SERRANO SIMÓN
21 de Febrero de 2014

RESUMEN

La finalidad de este proyecto es la implementación del método de identificación de sistemas dinámicos denominado test de relé modificado mediante una interfaz gráfica en Matlab. Se empleará su herramienta GUIDE para el desarrollo de la misma, de manera que se ofrezcan al usuario las funcionalidades necesarias que permitan la identificación de sistemas dinámicos tanto de velocidad como de posición a la frecuencia deseada, así como la configuración de los diversos parámetros del test.

Se desarrollará un manual de ayuda para el uso de la interfaz de identificación. Tanto la interfaz como su manual de ayuda se elaborarán en inglés para un mayor acercamiento a la comunidad científica internacional, entre la que se pretende difundir dicha herramienta software.

ABSTRACT

The purpose of this project is to implement a graphical user interface in Matlab (from now on GUI) for the identification of dynamic systems and using the modified relay test method. For the development of this GUI we use GUIDE toolbox, developing an interface that offers all the functionalities to allow the identification of velocity and position dynamic systems at a frequency given by the user, as well as the configuration of the different parameters of the relay test.

A manual is also going to be developed for the GUI. Both the manual and the GUI are in English to get closer to the international scientific community, which the aim of spreading this software tool across it.

1. INTRODUCCIÓN

1.1. Introducción

A causa de la gran importancia que los sistemas dinámicos tienen en el medio ambiente que nos rodea, los métodos de identificación adquieren un papel notable en diferentes áreas de conocimiento, tales como ingeniería de control, biomedicina, electrónica, etc... donde se necesita un modelo con gran precisión para fines como análisis, simulación, diseño, control y predicción.

Debido a que las técnicas actuales de control, en ocasiones pueden resultar bastantes complejas y poco económicas, se recurre a la identificación de estos sistemas dinámicos, mediante diferentes métodos que te permiten obtener de manera más sencilla y con bastante exactitud el modelo matemático deseado [6]. Entre todos los que pueden existir, este trabajo se centra concretamente en el método del test de relé, en una de sus versiones modificadas.

Con la intención de que el uso de este método sea utilizado de una manera rápida y cómoda, se propone la implementación de este en una interfaz gráfica a través de funciones y objetos gráficos.

El origen de las interfaces gráficas se remonta a un período donde la mayoría de ordenadores, para su uso requerían un mínimo de conocimientos técnicos si se querían usar como algo más que una consola para jugar. Tras la creación de las interfaces se evitó que el usuario tuviese que aprender un entorno complejo, de esta manera, el uso del ordenador se incrementó en la sociedad. Tanto es así, que en tiempos actuales podemos decir que las interfaces gráficas poseen un papel relevante en el mundo tecnológico [4] [5].

En función del uso que se las quiera dar, existen múltiples lenguajes para crearlas. Después de hacer un estudio sobre estos lenguajes, podemos destacar cuales son los principales: C++, Java, Matlab. Cualquiera de los tres resulta muy potente y de gran importancia en la actualidad, pero tras evaluar las necesidades del método a implementar y debido a que se el sistema requiere una simulación se opta por la tercera opción, Matlab.

Se trata de un programa muy utilizado en el mercado gracias a su lenguaje de alto nivel y su entorno interactivo para el cálculo numérico [11]. También se caracteriza por tener gran variedad de aplicaciones entre la que destacamos GUIDE, aplicación usada para la creación de interfaces gráficas y que emplearemos en este proyecto para la implementación del método de identificación de sistemas dinámicos mediante el test del relé modificado.

1.1. Objetivos

El trabajo fin de grado tiene como objetivo principal el diseño y la creación de una interfaz gráfica en MATLAB mediante la herramienta GUIDE, sobre la que se debe integrar el método de identificación de sistemas dinámico del test de relé.

Se pretende, que la nueva herramienta creada interactúe con el usuario y permita:

- Simular un sistema dinámico de posición o velocidad.
- Elegir la frecuencia la que se desea identificar el sistema dinámico.
- Después de realizar el test del relé, comparar los resultados experimentales y teóricos.
- Visualizar los resultados a través de gráficas.
- Almacenar datos, gráficas e imágenes del programa.
- Abrir un manual de ayuda para el usuario.

1.2. Estructura de la memoria

En este apartado, se muestra un resumen todos los capítulos que componen el trabajo fin de grado.

- *CAPÍTULO 1 "INTRODUCCIÓN":* Se expone una breve introducción acerca del contenido del documento, así como sus objetivos. También muestra la estructura que forma el trabajo fin de grado.

- *CAPÍTULO 2 "IDENTIFICACIÓN DE SISTEMAS DINÁMICOS MEDIANTE EL TEST DE RELÉ":* Se describe el método utilizado para la identificación del sistema dinámico.

- *CAPITULO 3 "IMPLEMENTACIÓN DEL TEST DE RELÉ EN MATLAB":* Se explica toda la programación necesaria para poder implementar en Matlab el método del test de relé correctamente, tanto para la identificación de un sistema de posición como de velocidad.

- *CAPÍTULO 4 "DISEÑO DE LA INTERFAZ GRÁFICA EN MATLAB":* Describe por un lado los aspectos más importantes del entorno MATLAB y por otro lado se centra en la descripción de su herramienta GUIDE para la creación de interfaces gráficas.

- *CAPÍTULO 5 "PROGRAMACIÓN DE LA INTERFAZ":* Se detalla toda la programación de la nueva interfaz creada, describiendo su estructura general y las funciones utilizadas.

- *CAPÍTULO 6 "FUNCIONAMIENTO DE LA INTERFAZ":* Se explica paso a paso el funcionamiento de toda la interfaz gráfica, usando como ejemplo la identificación de un sistema de posición.

- *CAPÍTULO 7 "ENTORNO SOCIO-ECONÓMICO":* Se incluye un presupuesto detallado de todos los conceptos del proyecto.

- *CAPÍTULO 8 "CONLUSIONES":* Se hace un análisis crítico sobre los resultados del trabajo.

- *CAPÍTULO 9 "TRABAJOS FUTUROS":* Se comentan las posibles propuestas para la mejora del trabajo.

- *CAPÍTULO 10 "REFERENCIAS":* Se muestran las fuentes consultadas a lo largo de todo el proyecto.

- *CAPÍTULO 11 "ANEXO-DOCUMENTO DE AYUDA":* Se incluye la documentación de ayuda en inglés que irá incorporada en la interfaz y todo el código que compone la interfaz.

2. IDENTIFICACIÓN DE SISTEMAS DINÁMICOS MEDIANTE EL TEST DE RELÉ

2.1. Introducción

Con frecuencia algunos modelos dinámicos no pueden conocerse mediante consideraciones teóricas, por lo que se recurre a determinarlos experimentalmente. La determinación de modelos dinámicos sobre la base de las mediciones realizadas durante un proceso, se conoce como identificación de sistemas dinámicos o identificación del proceso [3]. Entre los diversos métodos para la identificación que existen, este trabajo se centra en uno de ellos denominado test del relé modificado.

2.2. Test de relé

Se trata de un método de identificación que, partiendo de dos datos tales como la frecuencia y la amplitud del ciclo límite que genera, permite obtener la respuesta frecuencial del sistema a la frecuencia de oscilación, específicamente la magnitud y fase del sistema a esa frecuencia.

Consiste en un test que se realiza en bucle cerrado, y dependiendo de los parámetros del relé por los que se opte, el proceso puede ser llevado hasta un estado de oscilación de interés del sistema [8]

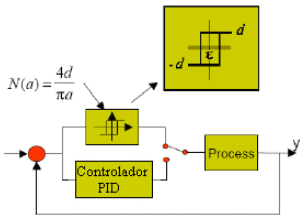


Figura 1: Esquema del test del relé

En la figura 1 se muestra un esquema de test del relé donde *N(a)* es la función descriptiva de un relé ideal sin histéresis. El sistema oscila ante ausencia de entrada (oscilador).

Se da por supuesto que hay un periodo de ciclo límite T_u ($\omega_u = \frac{2\pi}{T_u}$) de tal modo que la salida del relé sea una onda cuadrada simétrica de dicho período. El desarrollo en serie de Fourier para esta salida con $\varepsilon = 0$ (sin histéresis) da un primer armónico de amplitud $\frac{4d}{\pi}$. Se asume que el proceso dinámico tiene carácter de filtro paso bajo y que la salida viene dominada por el primer armónico, por lo que la señal de error tendrá la siguiente amplitud:

$$a = \frac{4d}{\pi}\left|G\left(j\frac{2\pi}{T_u}\right)\right|.$$

La condición de oscilación será:

$$\arg\left(G\left(j\frac{2\pi}{T_u}\right)\right) = -\pi,$$

$$\left|G\left(j\frac{2\pi}{T_u}\right)\right| = \frac{\pi a}{4d} = \frac{1}{N(a)}$$

Este punto de la planta es el mostrado en la figura 2:

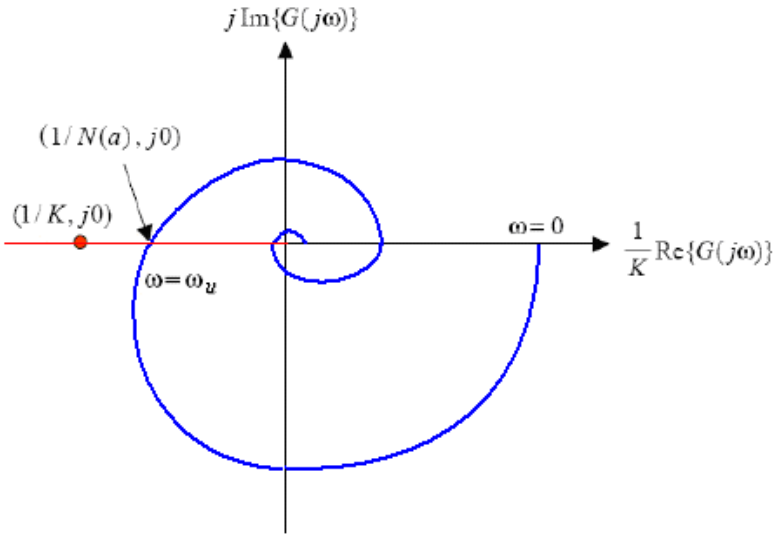


Figura 2: Punto de la curva de Nyquist de la planta obtenido con el test del relé

Definiendo:

$$K_u = \frac{4d}{a\pi},$$

es fácil de ver que, con los supuestos anteriores, K_u es la ganancia que lleva al sistema a una estabilidad límite bajo un control proporcional puro. Tanto la ganancia como el período límite son fáciles de determinar mediante el test del relé [4].

Sin embargo, mientras que este método resulta muy útil en la mayoría de los casos, presenta principalmente los siguientes inconvenientes:

- Debido a la suposición de comportamiento de filtro paso bajo de la planta y a las simplificaciones correspondientes, la estimación del punto crítico puede no ser suficientemente precisa. Bajo ciertas condiciones de procesos de orden elevado o de procesos con tiempos muertos altos, este método puede arrojar errores considerables de estimación.

- Con este método, sólo se obtiene el punto de respuesta a la frecuencia última ω_n, lo que puede resultar insuficiente, como el que nos ocupa.

Para solucionar este último problema, se puede usar una variación del test del relé, como se muestra en la figura 3.3:

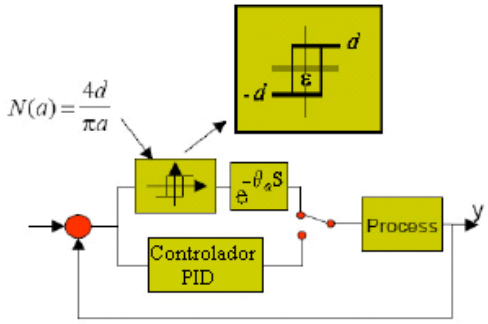


Figura 3: Esquema del test del relé modificado

Se añade un tiempo de retardo en el lazo directo para cambiar la frecuencia de oscilación del relé. Para cada valor de θ_a, se obtiene un punto diferente en la curva de Nyquist de la planta. Así, se puede identificar un punto de la curva de Nyquist a cualquier frecuencia deseada, por ejemplo, la frecuencia de corte ω_{cg} que se desee especificar par el sistema.

La fase y magnitud de este punto vienen definidas por las siguientes expresiones:

$$\arg\left(G\left(j\,\omega_{cg}\right)\right) = -\pi + \omega_{cg}\theta_a,$$

$$\left|G\left(j\,\omega_{cg}\right)\right| = \frac{\pi\,a}{4\,d} = \frac{1}{N(a)}$$

La siguiente figura muestra la curva de Nyquist resultante.

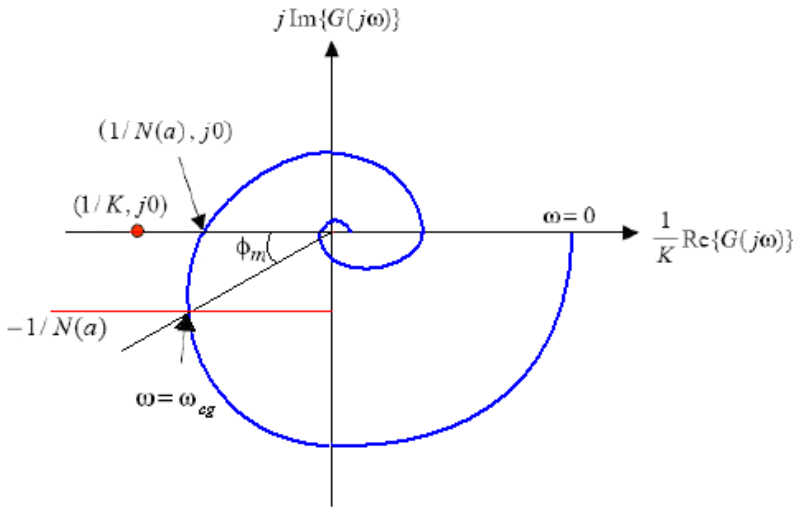


Figura 4: Diagrama de Nyquist para la función con retardo

El problema sería obtener el valor correcto de θ_a que produce una oscilación a la frecuencia de corte ω_{cg} deseada. Un método iterativo puede utilizarse para resolver este problema:

1. La frecuencia ω_{cg} y δ se fijan al inicio del proceso. Se define ω_{cg} como la frecuencia a la que se desea oscilar y δ como el error máximo permitido en dicha frecuencia.

2. Hay que seleccionar correctamente dos valores iniciales de retardo (θ_{-1} y θ_0) y realizar el test del relé para cada uno de ellos. Así, se obtendrán dos puntos en la curva de Nyquist. La frecuencia de estos puntos se puede representar como ω_{-1} y ω_0, que se corresponden con θ_{-1} y θ_0 respectivamente. El ciclo de repeticiones de este método comienza precisamente con estos valores iniciales [$(\theta_{-1}, \omega_{-1})$ y (θ_0, ω_0)].

3. Con los valores obtenidos en el punto anterior, el valor de retardo siguiente se calcula de forma automática según la siguiente fórmula de interpolación / extrapolación:

$$\theta_n = \frac{\omega_{cg} - \omega_{n-1}}{\omega_{n-1} - \omega_{n-2}}(\theta_{n-1} - \theta_{n-2}) + \theta_{n-1},$$

donde *n* representa el número actual de iteraciones. Para el nuevo valor θ_n se vuelve a realizar el test y se memoriza la frecuencia de oscilación resultante, ω_n.

4. Se compara ω_n con ω_{cg}. Si $|\omega_n - \omega_{cg}| < \delta$, se sale del bucle. En caso contrario, se repite el paso número 3. δ es un número pequeño positivo.

Este método iterativo es factible porque para la mayoría de las plantas la relación entre el tiempo de retardo θ_n y la frecuencia de oscilación ω_n, es de uno a uno [8].

3. IMPLEMENTACIÓN DEL TEST DE RELÉ EN MATLAB

Tras comprender la explicación teórica expuesta en apartado 2 sobre el método test de relé, a continuación se pasa a su traducción en el lenguaje MATLAB. Debido a que los pasos del proceso de identificación son bastante semejantes para ambos sistemas, la explicación de la programación se centra en un sistema de posición, no olvidando citar las posibles diferencias que tiene el sistema de velocidad.

Para comenzar con el proceso, al no tener ningún sistema físico real, resulta imprescindible utilizar alguna herramienta que simule el sistema que se quiere identificar, para ello se utilizar Simulink, otra de las aplicaciones de Matlab, que a través de diferentes diagramas de bloques, permite su simulación.

La figuras 5 y 6 respectivamente, ofrecen el diseño del modelo explicado en el apartado 2.2, para un sistema de posición y de velocidad.

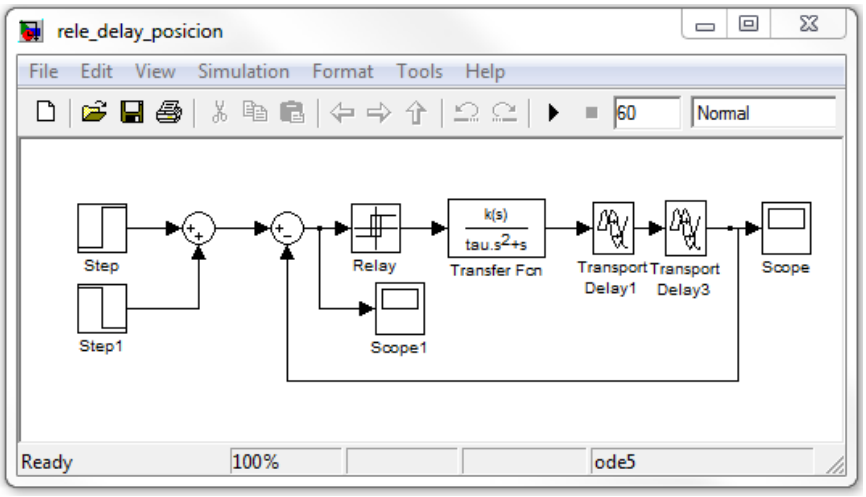


Figura 5: Simulación de un sistema de posición.

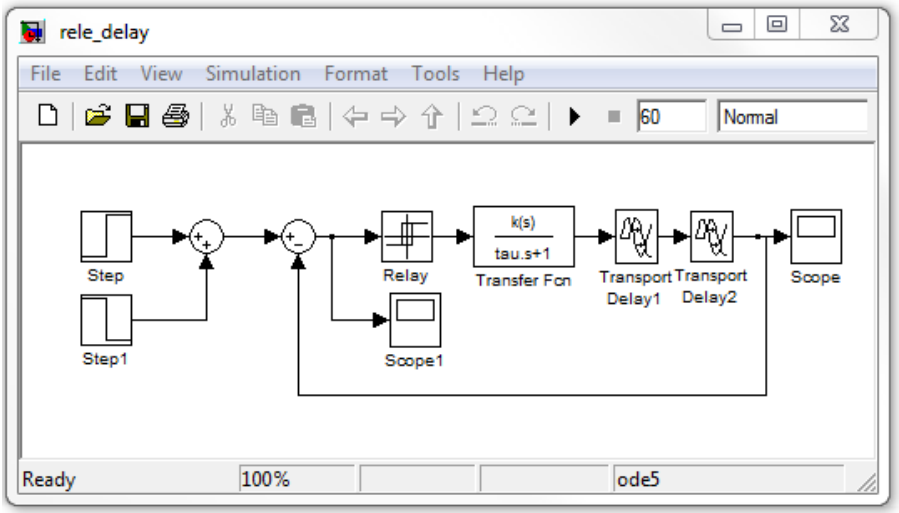


Figura 6: Simulación de un sistema de velocidad.

Dentro de estos modelos se puede observar el uso de variables tales como *k,tau,d,L_rele,* cuyo valor es definido en el apartado 3.1.

Para poder seguir con el método,se utilizan funciones encargadas de completar el proceso. A continuación se detallan algunos aspectos importantes de estas funciones.

3.1. Implementación del test de relé

Es una función encargada de realizar todo el proceso de identificación del sistema dinámico. Recibe unas variables de entrada procedentes de otra función, así como devuelve unas variables de salida.

```
function[amplitud,periodo_error,wu,L_rele,magnitud_estimada,magnit
ud_real,arg_estimado,arg_real]=
Rele_Param_Delay_Pos(k,tau,L,wc,rango,d,...
L_rele_1,L_rele_2,g_iteracion,grafica_1,grafica_2)
```

Para evitar problemas al cargar las variables necesarias en el archivo de simulink, estas variables se deben guardan en el "*Workspace*" (espacio de trabajo de Matlab), para ello se hace uso de la sentencia *assignin.*

```
%L_rele=rele de la planta
L_rele=L_rele_1;
assignin('base','L_rele', L_rele);
%wc=frecuencia de corte deseada
assignin('base', 'wc', wc);
%k=ganancia
assignin('base','k', k);
%tau=cte. de tiempo;
assignin('base','tau', tau);
%L=retardo planta;
assignin('base','L', L);
%d=amplitud del rele;
assignin('base', 'd', d);
%rango=rango de frecuencias;
assignin('base', 'rango', rango);
```

En el caso de que el valor de la variable cambie a medida que avanza el proceso (como ocurre con la variable de retardo del relé), se deben hacer unos pasos intermedios. Primero se asigna el valor *L_rele_1* que entra en la función a una nueva variable local *L_rele* y a continuación se guarda en el "*Workspace*".

```
%L_rele=rele de la planta
L_rele=L_rele_1;
assignin('base','L_rele', L_rele);
```

Cuando llega al punto donde el valor del retardo debe cambiarse por la otra variable *L_rele_2* correspondiente al segundo relé, de nuevo se asigna el valor a esta misma variable *L_rele* y se cargar en el "*Workspace*".

```
L_rele=L_rele_2;
assignin('base', 'L_rele', L_rele);
```

A continuación, se llama al archivo simulink *rele_delay_posicion.mdl* donde se cargan todas las variables y comienza la primera simulación.

```
sim('rele_delay_posicion')
```

Con la intención de indicar al usuario un estado de espera mientras que se realiza todo el proceso, se utiliza la sentencia *waitbar* .Muestra sobre una ventana un mensaje y una barra de espera donde su duración depende del bucle *while* (Ver figura 17).

```
bp=waitbar(0,'Program in execution. Please,wait...','Name','');

M=40;
I=0;

while I<M

I=I+1;
H=2*I;
G=I+3*I;
```

```
waitbar(I/M,bp);

end

    delete(bp)
```

Cuando todo esta listo, se cierra la ventana y se entra en la parte importante del método del test de relé modificado.

Al haber utilizado una entrada escalón para excitar el sistema, para visualizar los resultados sobre 0 y no sobre la entrada escalón, se elimina la referencia escalón de la señal de salida.

```
relesinref=[rele(:,1),rele(:,2)];
```

Para que los resultados tengan una mayor precisión, se utilizan únicamente los últimos 60 puntos de salida, donde la señal ha llegado a su estado permanente.

```
t=relesinref(60:length(relesinref),1);
error=relesinref(60:length(relesinref),2);
```

En la siguiente variable se almacena el máximo valor de la señal de salida.

```
amplitud_max=max(error)
```

A continuación, se realizar un proceso de iteración donde se calcula la amplitud y frecuencia a partir de señal anterior.

```
pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
    if sign(error(ind)) ~= sign(anterior)

        if cont==0
        error1=error(ind:length(error));
        amplitud_min=min(error1)
        amplitud=(amplitud_max-amplitud_min)/2
        cont=1;
        end
```

```
        if error(ind) > 0
            pos=[pos ind];
        else
            pos=[pos ind-1];
        end
    end
    anterior=error(ind);
end
pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
    periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
    cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error
```

Para ver la evolución del proceso, se pinta el proceso sobre una gráfica mediante la sentencia *plot*.

Con la intención de dibujar las gráficas en el mismo panel que se le ha destinado en la función *main*, se utiliza la variable de entrada *g_interacion* .

```
plot(g_iteracion,t,error,'.',t(pos),error(pos),'or'),grid on;
title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
```

Mediante *ylabel* y *xlabel* se pintan los títulos correspondientes sobre cada eje.

```
ylabel(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,
'Fontname','Courier New');
xlabel(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,
'Fontname','Courier New');
grid (g_iteracion);
```

Para que estas gráficas puedan ser almacenadas automáticamente, se procede a repetir el código con unas pequeñas modificaciones.

```
figura=figure
subplot(5,1,1)
plot(t,error,'.',t(pos),error(pos),'or'),grid on;
ylabel('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname',
'Courier New');
```

```
xlabel('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname',
'Courier New');
grid on;
```

La declaración de una nueva figura sirve para dibujar sobre ella las diferentes gráficas que se quieren almacenar.

La sentencia *subplot (m,n,p)* divide la cifra actual en una rejilla de m-por-N y crea un ejes en la posición de cuadrícula especificada por p. La primera cuadrícula es la primera columna de la primera fila, la segunda rejilla es la segunda columna de la primera fila, y así sucesivamente [10].

Con el nuevo valor de frecuencia obtenido y el valor del primer retardo, actualizamos los vectores correspondientes .Se solicita un segundo valor de retardo *L_rele_2*, y se repite de nuevo el proceso de iteración anteriormente explicado.

```
w_total=[w_total wu]
L_total=[L_total L_rele]
L_rele=L_rele_2;
assignin('base', 'L_rele', L_rele);
```

Tras finalizar el proceso, se evalúa que mientras el nuevo valor de la frecuencia no se aproxime a la frecuencia de corte elegida por el usuario en menos de 0.15, el programa calcula iterativamente nuevos valores de retardo y sus frecuencias correspondientes.

```
while(abs(wu-wc)>0.15)
{…}
End
```

Si esta condición se cumple se sale del bucle, se almacena el último valor de retardo *L_rele* y la última frecuencia *wu* que este retardo ha ocasionado. A continuación se pasa a calcular la magnitud y fase del sistema, para ello se llama a una nueva función *Magnitud_Delay_Int* .

```
[magnitud_estimada,magnitud_real,arg_estimado,arg_real]=Magnitud_D
elay_Int(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rele,grafica_
1,grafica_2,figura);
```

En el caso de que el sistema a simular sea de velocidad se hace el mismo procedimeinto de las dos funciones, aunque cambiaría el nombre de la función *Rele_Param_Delay* y la llamada al archivo de simulink *rele_delay.mdl* (Ver apartado 11.1.2).

```
function[amplitud,periodo_error,wu,L_rele,magnitud_estimada,magnit
ud_real,arg_estimado,arg_real]=Rele_Param_Delay(k,tau,L,wc,rango,d
,L_rele_1,L_rele_2,g_iteracion,grafica_1,grafica_2).

sim('rele_delay')
```

3.2. Estimación de las características frecuenciales del sistema

Se trata de una función encargada de dibujar el diagrama de Bode de la planta real y no de la estimada.

```
function[magnitud_estimada_db,magnitud_real_db,argumento_estimado,
argumento_real]=
Magnitud_Delay(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rele,gr
afica_1,grafica_2,figura)
```

A partir de valores de entrada se crea la función de transferencia de la planta.

```
rango=2;
w=logspace(-rango+log10(wc),rango+log10(wc),200);
ret_real=exp(-L*j*w);
num_real=k*ret_real;
den_real=polyval([tau 1],(j*w));
total_real=num_real./den_real;
mgtotal_real=20*log10(abs(total_real));ftotal_real=(180/pi)*angle(
total_real);
```

A continuación se pasa a dibujar las gráficas correspondientes al diagrama de bode.

```
subplot(5,1,4)
semilogx(w,mgtotal_real);
title ('Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','CourierNew')
;
ylabel('Magnitude
(dB)','Fontweight','Bold','Fontsize',11,'Fontname','Courier New');
 grid on;
subplot(5,1,5)
semilogx(w,ftotal_real)
 grid;
 xlabel('Frequency
(rad/s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
 ylabel('Phase
(deg)','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
 grid on;
 saveas(figura,'graficas.fig')
 set(figura,'visible','Off')
```

Se puede observar como la programación de las gráficas resulta prácticamente igual que las explicadas en el apartado 3.1, con algunas diferencias como la nueva sentencia *semilogx*, encargada de dibujar el eje x con escala logarítmica.

Además las dos últimas líneas resultan de gran importancia ya que son las encargadas de guardar la figura creada para las gráficas en un archivo de tipo *fig*. Para que esta figura no se superponga a la figura principal basta con añadir el comando *set*.

```
saveas(figura,'graficas.fig')
set(figura,'visible','Off')
```

En esta función de nuevo, no se puede apreciar ningún cambio a nivel de programación con respecto al sistema de velocidad, más que la declaración de la función.

```
function[magnitud_estimada_db,magnitud_real_db,argumento_estimado,
argumento_real]=
Magnitud_Delay_Int(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rel
e,grafica_1,grafica_2,figura)
```

Después de haber integrado el test de relé correctamente en Matlab y continuando con los objetivos del trabajo, se pasa a la conexión de este método con la interfaz gráfica, en el apartado 5 se entra en más detalle.

4. DISEÑO DE LA INTERFAZ GRÁFICA EN MATLAB

4.1. MATLAB

4.1.1. Introducción

MATLAB® es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación. Mediante MATLAB, es posible analizar datos, desarrollar algoritmos y crear modelos o aplicaciones. El lenguaje, las herramientas y las funciones matemáticas incorporadas permiten explorar diversos enfoques y llegar a una solución antes que con hojas de cálculo o lenguajes de programación tradicionales, como pueden ser C/C++ o Java™ [1].

Así, es usado en una amplia variedad de aplicaciones, incluyendo procesamiento de señal e imagen, comunicaciones, diseño de control, biología computacional, análisis financiero, estadística, álgebra lineal, etc. Existen diversas *toolboxes* específicamente diseñadas para las distintas áreas de aplicación, que contienen funciones especiales para resolver los problemas propios de estas[2].

El nombre MATLAB deriva del término inglés *'matrix laboratory'*, ya que la principal característica de MATLAB es que permite manipular de manera sencilla y eficiente vectores y matrices [1].

Existe una versión MATLAB ® Mobile ™ disponible para dispositivos iPhone, iPad o Android ™ que se conecta a una sesión de MATLAB ® la cual se ejecuta en la nube MathWorks o en su computadora. Desde el dispositivo móvil, puede ejecutar scripts, crear y manipular cifras, y ver los resultados [12].

El sistema MATLAB está constituido por estas partes principales [2]:

- **Herramientas del escritorio y Entorno de desarrollo:** esta parte de MATLAB es el conjunto de herramientas que facilitan el trabajo con archivos y funciones. Muchas de estas herramientas son interfaces gráficas de usuario. Éstas incluyen: el escritorio MATLAB y la ventana de comandos, un editor y un depurador de código, así como exploradores para visualizar el contenido de ayuda, el entorno de trabajo y las carpetas.

- **Librería de funciones matemáticas:** esta librería es una amplia colección de algoritmos computacionales que van desde funciones elementales como suma, seno, coseno y aritmética compleja, hasta funciones más sofisticadas como la matriz inversa, cálculo de los autovalores de una matriz, funciones de Bessel o la transformada rápida de Fourier.

- **El lenguaje:** el lenguaje MATLAB es un lenguaje de alto nivel construido con matrices y vectores, con sentencias de control de flujo, funciones, estructuras de datos, y funcionalidades de entrada y salida y programación orientada a objetos.

- **Gráficos:** MATLAB cuenta con amplias instalaciones para la visualización de vectores y matrices como gráficos, además permite guardar e imprimir cada uno ellos. Incluye un alto nivel de funciones para la visualización de datos en dos dimensiones y en tres dimensiones, gráficos de procesamiento de imágenes, animación y presentación. También contiene funciones de bajo nivel que permiten personalizar totalmente la apariencia de gráficos, así como construir interfaces gráficas de usuario para aplicaciones de MATLAB.

- **Interfaces externas:** La biblioteca de interfaz externa permite escribir programas en C y Fortra. Esta incluye instalaciones para llamar a rutinas de MATLAB (enlace dinámico), utilizar a MATLAB como motor de cálculo, y para la lectura y la escritura MAT-archivos.

A continuación se explican algunos aspectos fundamentales del lenguaje Matlab, para facilitar la interpretación de la programación de los apartados 3 y 5. También se detallan algunos aspectos generales sobre la herramienta GUIDE usada para crear la interfaz.

4.1.2. Variables en MATLAB

MATLAB permite asignar nombres variables a valores numéricos. Los nombres de las variables deben empezar con una letra seguida de una serie de caracteres alfanuméricos. Una apreciación sobre este lenguaje es que distingue entre letras mayúsculas y minúsculas. Los nombres de las variables pueden tener toda la longitud que se quiera, pero MATLAB sólo usará los primeros N caracteres e ignorará el resto. Es importante dentro asegurarse que dentro de una misma función dos variables no estén definidas con el mismo nombre.

La sentencia para la creación de la variable es de la forma:

```
>> variable = expresion
```

En “expresion” pueden aparecer principalmente: matrices, valores numéricos, otras variables y operadores.

Las **matrices numéricas** rectangulares están formadas por números reales o complejos. De este modo, un escalar en MATLAB es una matriz de dimensiones 1x1, y un vector es una matriz de una única fila o columna [1].

El siguiente ejemplo muestra la declaración de una matriz:

```
>> Matriz = [1 2 3; 4 5 6]
Matriz =
1 2 3
4 5 6
```

Una **matriz de tipo estructura** es un conjunto heterogéneo de datos distribuidos en distintos grupos llamados campos. Para crear un nuevo campo, o modificar el valor de uno existente se utiliza la siguiente sentencia:

```
>> estructura.campoX= 'nuevo texto'
```

Dentro de las **variables de tipo numérico**, existen diferentes tipos, se pueden destacar datos de tipo flotante (float), entero (int), de doble precisión (double), etc…

También existen **variables de texto**. Para asignar el valor a la variable se debe introducir el texto entre comillas, quedando su declaración de la siguiente manera:

>> Variable_de_texto = “Texto”

Este tipo de variables son en realidad un vector fila, quedando almacenado cada carácter de texto en una componente del vector. Las variables de texto se pueden convertir en numéricas. En función de la precisión con la que la que se quiere realizar la conversión tenemos, por un lado la sentencia *str2num* y para una mayor precisión en su resultado tenemos la sentencia str2double .Para el proceso inverso donde se convierte una variable numérica en una variable de texto, solo existe una única sentencias *num2str*.

Por defecto, MATLAB utiliza la doble precisión para almacenar variables numéricas.

En función de la operación que se desee realizar tenemos por una parte, para expresar condición los operadores relacionales: igual “==”, menor o igual “<=”, mayor o igual “>=”, menor “<”, mayor “>”, distinto “~=”, y los operadores lógicos: AND “&”, NOT “~” y OR “|” y por otra parte tenemos los operadores aritméticos ya conocidos.

4.1.3. Tipos de variables

En función del uso que se le quiera dar a la variable dentro del programa, MATLAB divide las variables en tres tipos de: locales, globales y persistentes [4].

Variables locales: se trata de todas las variables que son definidas dentro de cada función, estas no permanecen en memoria una vez que se llama a una nueva función.

Variables globales: se trata de variables que son compartidas por diferentes funciones, para ello en cada función donde se quiere usar, se debe declarar la variable con el prefijo *GLOBAL.* Para identificarlas mejor es recomendable escribirlas en letras mayúsculas.

Variables persistentes: se trata de variables que sólo pueden ser declaradas en funciones, y sólo tendrán acceso a ellas las funciones donde han sido declaradas. Cuando termina la ejecución de una función, MATLAB no borra estas variables de la memoria, así su valor persiste en las sucesivas llamadas a la función.

4.1.4. Alcance de una variable

MATLAB almacena las variables en un espacio de memoria llamado *workspace* o espacio de trabajo. Las variables creadas de manera interactiva y las creadas al ejecutar scripts se almacenan en el *espacio base de trabajo.* Pero las funciones no usan este *espacio base de trabajo,* porque cada función tiene su propio espacio de trabajo. A continuación se muestran algunas de las opciones para ampliar el alcance de una variable:

- Usar variables globales

- Pasar la variable a la función que necesita usarla, como un argumento extra en la llamada a la función.

- Uso de funciones anidadas: el alcance de una variable local se extiende a todas las funciones que estén anidadas dentro de ella.

Para eliminar los valores de variables almacenadas en el espacio de trabajo, se emplea el comando *clear* seguido del nombre de la variable. Para eliminar los valores de todas las variables, se emplea el comando *clear all*.

4.1.5. Programación con MATLAB

La programación con MATLAB es muy similar a la realizada con otros lenguajes de programación. Podemos encontrar las sentencias habituales para el control condicional (sentencias if/ else/ elseif, y sentencias *switch/ case/ otherwise*) y el control por ciclos (*for, while, continue* y *break).*

4.1.5.1. Tipos de archivos

Aunque MATLAB puede usarse de un modo interactivo, su utilización más habitual consiste en la ejecución de secuencias de sentencias que deberán almacenarse en archivos. En este sentido trabaja de manera similar a cualquier lenguaje de programación [1].

Estos archivos se conocen como *M-File* y tienen extensión ".m". Son compilados de manera automática durante su primera ejecución. Hay dos tipos de ficheros *M-File*: los *ficheros script* y las *funciones*.

Los **ficheros script** consisten en una serie de sentencias y comandos de MATLAB. Cuando se ejecuta un fichero script, todas las variables utilizadas en el mismo quedan almacenadas en memoria durante la sesión actual de trabajo.

Las **funciones** deben siempre comenzar con la siguiente sintaxis:

function [argumentos de salida] = nombre_funcion (argumentos de entrada)

Otro requisito que debe cumplir la función, es que el nombre del archivo debe ser igual que el nombre de la función en la declaración de la primera línea. En el ejemplo de arriba, el archivo se llamaría "nombre_funcion.m".

La diferencia principal entre una función y un script es que las variables que se definen y manipulan dentro de la función sólo permanecen en memoria mientras dura la ejecución de la misma; por tanto, las únicas variables que permanecen en memoria en la sesión de trabajo son los argumentos de entrada y salida.

Para interrumpir la ejecución de un script o una función, se emplea la sentencia *return.*

Los archivos de tipo función pueden contener funciones adicionales con nombres diferentes, que se irán escribiendo unas a continuación de otras. La primera función es la *función principal* y las demás son ***subfunciones*.** Cada subfunción debe comenzar con su propia sentencia de declaración de función.

4.1.5.2. Sintaxis de las funciones

En la declaración de una función, si no hay argumentos de salida se pueden omitir los corchetes y el signo "=". Si sólo hay un argumento de entrada no es necesario corchetes, y si no hay argumentos de entrada no es necesario poner el paréntesis.

Si el número de argumentos de la función es variable, se pueden emplear las variables *varargin* y *varargout*.
function varargout = nombre_funcion (varargin)

La variable *varargin* es una matriz de celdas que contiene todos los argumentos opcionales de la función. Recoge todos los argumentos que se introducen al producirse la llamada a la función. La variable *varargout* es una matriz de celdas que recoge todos los argumentos de salida a los que se haya asignado valores dentro de la función.

Siguiendo con el ejemplo anterior, podríamos extraer las variables que nos interesan de la matriz *varargin* del siguiente modo.

```
[variable_1, variable_2] = varargin {[2,5]};
```

Y podemos asignar valores a las variables de la matriz *varargout* que consideremos.

```
varargout {1} = resultado;
```

También existe un modelo de función utilizada en la programación de la interfaz, su sentencias es la siguiente [13]:

```
function pushbutton1_Callback(hObject,eventdata,handles)
```

Es un tipo de función usada para declarar el funcionamiento tras ejecutar una acción a través de un uicontrol, tienen los siguientes argumentos de entrada estándar:

hObject – se trata del componente de interfaz gráfica de usuario, por lo que se activó la devolución de llamada. Para un grupo de botones de SelectionChangeFcn devolución de llamada, hObject es la manija del botón de opción seleccionado o botón de alternar.

EVENTDATA - secuencias de eventos provocados por las acciones del usuario, tales como selecciones.

handles - estructura de MATLAB que contiene todos los objetos de la interfaz gráfica de usuario, y también puede contener datos definidos por la aplicación.

4.1.6. GUIDE

Con la intención de crear una interfaz gráfica de usuario, del inglés *GUI* (*'Graphical user interface')* se recurre a una aplicación propia de MATLAB denominada GUIDE (GUI Development Enviroment), es una herramienta interactiva que facilita el diseño y la construcción de esta.

Al comenzar el proceso de creación de una GUI mediante esta herramienta, aparece una figura que podemos ir poblando de controles y objetos por medio de un editor gráfico. GUIDE crea un archivo de código asociado a esta figura, que contiene los *callbacks* de la GUI y sus componentes. GUIDE guarda ambos, la figura (como un archivo *.fig*), y el código (como un archivo *.m*). Al abrir uno de ellos se abre el otro para arrancar la GUI.

4.1.7. Alternativa para construir un GUI

Complementando el apartado 4.1.6, podemos decir que existe una alternativa para crear una GUI en MATLAB, mediante programación.
Se crea crear un archivo de código en el que se definen todos los elementos y comportamientos de la GUI. Al ejecutar el archivo, se crea una figura en la que aparecen todos los elementos programados.

Normalmente, la figura no se guarda entre sesiones porque el programa crea una nueva cada vez que se arranca el programa.
Este procedimiento resulta ser una menos práctico ya que se deben definir todas las propiedades de la figura y sus controles, así como los *callbacks,* cosa que la aplicación GUIDE ya los tiene definidos internamente.

4.1.8. Componentes de forman la GUI

En una primera aproximación, se puede decir que una *GUI* está basada en objetos gráficos. El objeto principal es una figura. De hecho, se puede decir que en MATLAB, una *GUI* es en sí una figura. Las figuras pueden contener los siguientes objetos gráficos [1]:

Objetos *axes*: definen una región dentro de la figura, en la cual se representarán imágenes, dibujos en dos o tres dimensiones, se escribirán textos, etc.

Objetos *uicontrol*: abreviatura del término inglés „*user interface control'*. Son cajas de texto y botones que permiten ejecutar una determinada acción, programada por el usuario, cuando se activan con el ratón.

Objetos *uimenu*: abreviatura del término inglés *user interface menu'*. Son los menús creados por el usuario que se añaden al menú ya existente en la parte superior de la figura.

Objetos *uicontextmenu*: abreviatura del término inglés *user interface context menu'*. Son menús que se activan con el botón derecho del ratón y que pueden asociarse a cualquiera de los objetos anteriores.

Para diseñar la GUI de una manera más organizada la se recurre a utiliza paneles, los cuales agrupan los distintos objetos gráficos, otros paneles, etc... , esto se hace mediante la función ***uipanel***.

4.1.9. Propiedades de los objetos gráficos

Los objetos gráficos tienen una serie de propiedades que determinan su aspecto en la pantalla. Algunas de las más importantes son su posición en la jerarquía de objetos gráficos (*parent, children*), su visibilidad (*on, off*), el color, la posición, etc. Si no se indican otros valores de manera explícita, estas propiedades tomarán los valores que el objeto gráfico tenga definidos por defecto.

A continuación se indican algunas funciones necesarias para interaccionar con las propiedades de los objetos gráficos [1]:

- get (h): donde "*h*" es el *handle* del objeto gráfico, permite obtener un listado de todas las propiedades del objeto gráfico.

- set (h): proporciona un listado por pantalla de todas las propiedades del objeto con los posibles valores que puede tomar cada propiedad.

- get (h,"propiedad"): permite conocer el valor de la propiedad "*propiedad"* del objeto "*h"*.

- set (h,"propiedad"): permite conocer los valores que puede tomar la propiedad "*propiedad"* del objeto "*h"*.

- set (h,"propiedad", nuevovalor): permite asignar el valor "*nuevovalor", a* la propiedad "*propiedad"* del objeto "*h"*.

4.1.10. Uicontrols

Como se comentó antes, los objetos *uicontrol* son cajas de texto y botones que permiten ejecutar acciones programadas por el usuario, cuando se activan con el ratón. Distinguimos los siguientes tipos de *uicontrol* [1]*:*

- Push button: al hacer clic en ellos se ejecuta una determinada acción.
- Toogle button
- Radio button
- Checkbox
- Edit text: cajas de texto editables.
- Static text: cajas de texto no editables.
- Slider: permite ejecutar acciones dependientes de un parámetro numérico cuyo valor varía al trasladar el botón a lo largo de una barra rectangular por medio del ratón.
- Listbox
- Popup menú
- Frame: marco cuya función es agrupar grupos de botones.

Los *uicontrols* de tipo *Toogle button*, *Radio Button* y *Checkbox* permiten realizar una acción al hacerse clic en ellos. Esta acción se mantiene hasta que se vuelve a hacer clic de nuevo en el mismo. Suelen emplearse para elegir entre opciones.

Los de tipo *Listbox* y *Popup Menu*, muestran una lista de alternativas que harán que se ejecute una determinada acción programada al seleccionar una de ellas con el ratón.

Se puede crear *uicontrols* por la función "uicontrol", cuya sintaxis es la siguiente:

h1 = uicontrol
(identiicadorpadre,"propiedad1","valor1","propiedad2","valor2",…)

donde *identificadorpadre* es el identificador de la figura a la que se añade el *uicontrol.*

Las propiedades más relevantes que presentan los *uicontrols* son las siguientes:

- BackgroundColor: es el color del objeto.

- Callback: especifica la acción a realizar por MATLAB al actuar sobre el botón con el ratón.

- Enable: permite desactivar el *uicontrol* de modo que no sea posible actuar sobre él con el ratón. Toma valores *on* y *off.*

- Position: determina posición y tamaño del *uicontrol* dentro de la figura. Se define mediante un vector de cuatro elementos: [x y anchura altura]. La propiedad "units" establece las unidades en que se mide la posición: *inches, centimeters, normalized, points, pixels* y *characters.*

- String: es el texto que aparece en el *uicontrol.* El tamaño, tipo, color y grosor de la fuente, el ángulo de la letra, pueden modificarse mediante las propiedades *FontSize, FontName, ForegroundColor, Fontweight y FontAngle* respectivamente.

- Style: es el tipo de *uicontrol*: pushbutton, tooglebutton, radio-button, checkbox, edit, text, slider, frame, listbox y popupmenu.

- Value: en los *uicontrols* de tipo Checkbox, Radio Button y Toogle Button toma el valor "1" al estar seleccionado, y el valor "0" en caso contrario.

- TooltipString: permite introducir un comentario que se mostrará en un recuadro amarillo al acercarnos con el ratón al *uicontrol.*

5. PROGRAMACIÓN DE LA INTERFAZ

5.1. Introducción

Debido a la complejidad de un primer diseño de la interfaz y por los conocimientos previos de otros lenguajes, en este trabajo fin de grado se opta por la opción programática, dejando a un lado la otra opción que está más encaminada a interfaces sencillas.

Para facilitar la explicación de la programación, se divide el programa en tres bloques generales, la figura 3 muestra los archivos que componen cada uno de ellos.

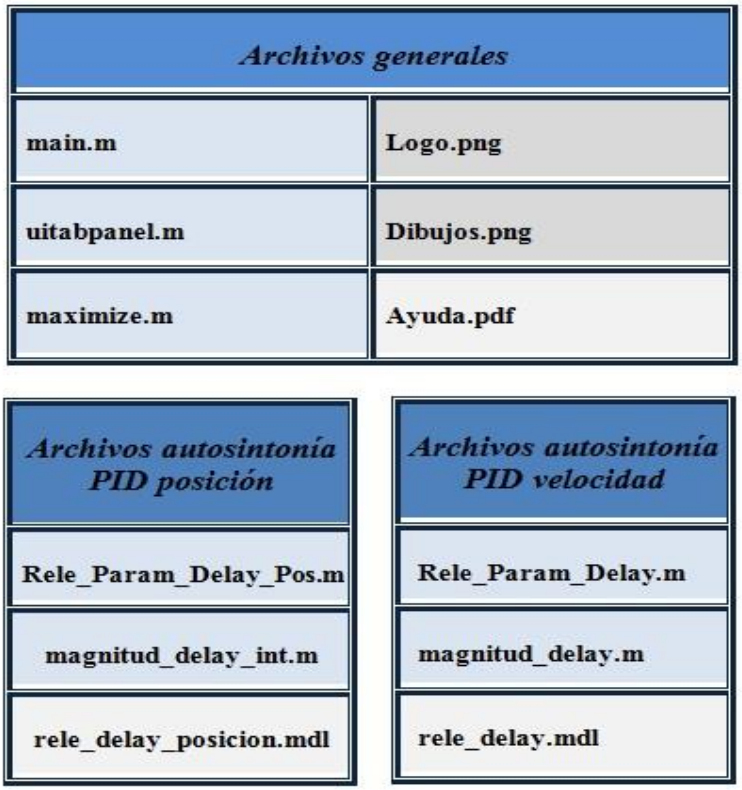


Figura 7: Estructura general del programa

5.2. Archivos generales

Entrando en el detalle de los archivos generales, se pueden distinguir en primer lugar el principal archivo del programa main.m. Se trata de un archivo compuesto por una función principal y unas subfunciones. En la Figura 8 se muestra un esquema general.

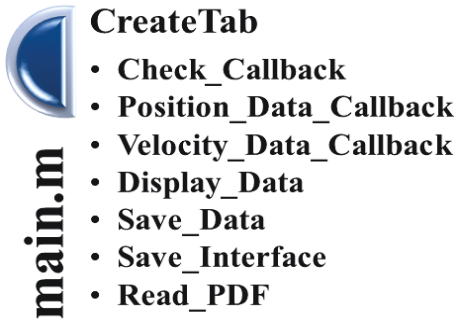


Figura 8: Esquema general de la función main

Con el objetivo de dar al interfaz un aspecto más complejo ,se recurre a dos funciones externas al programa, contenidas dentro de los archivos uitabpanel.m y maximize.m, ambas se detallan en el apartado 5.2.1 y 5.2.2.

Los últimos archivos tienen poco importancia a nivel de programación, son complemento para el diseño de la interfaz, serán llamados desde la función principal. Las figuras 9,10 y 11 muestran su contenido:

$$G(s) = \frac{k}{(\tau s + 1)s} e^{-Ls} \qquad G(s) = \frac{k}{(\tau s + 1)} e^{-Ls}$$

Figura 9: Archivo *Dibujos.png*

Figura 10: Archivo *Logo.png*

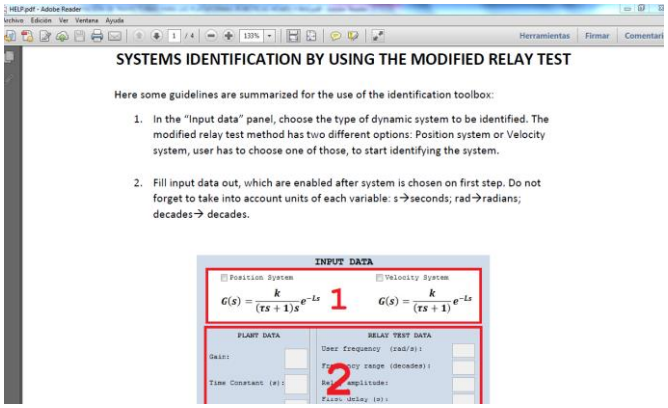


Figura 11: Archivo *Help.pdf*

5.2.1. Función uitabpanel

Con vistas a mejoras futuras del trabajo, se plantea crear diferentes pestañas dentro de la interfaz. Al no estar disponible esa opción en guide es necesario recurrir a una función externa *uitabpanel*. Esta permite crear pestañas y colocarlas en la posición deseada mediante el parámetro '*TabPosition*'.

Esta función es llamada dentro de la función *main*, para poder crear esta pestaña previamente se debe haber creado una figura interfaz sobre la que se van a pintar . Nuestra interfaz solo tiene una pestaña *IDENTIFICATION* [15].

```
uitabpanel(...
        'Parent',interfaz,...
        'TabPosition','lefttop',...
        'Units','Pixels',...
        'Position',[0,0,1.1*ANCHO,0.93*ALTO],...
        'PanelBorderType','line',...
        'Title',{'    IDENTIFICATION    ',},...
        'BackgroundColor',[0.078,0.169,0.549],...
        'CreateFcn',@CreateTab);
```

5.2.2. Función maximize

Esta función debido a su utilidad, ha sido recuperada de otro proyecto [7], igualmente puede descargarse de su correspondiente enlace [14].

Al hacer la llamada a esta función se maximiza la figura que se le pasa como parámetro de entrada .La sintaxis de la función es la siguiente:

>>maximize (figure)

Esta función es llamada desde el *main* (Ver apartado 5.2.3), para maximizar la pantalla temporal de espera. Y una vez cargados todos los controles de la interfaz, se llama a una nueva figura correspondiente a la pantalla principal y ésta también se maximiza.

5.2.3. Función main

A continuación se explican las funciones que componen el archivo principal main.m, para ver la programación entera consultar el apartado 11.1.5.

La función *main* es la encargada de construir la interfaz, está compuesta de una pantalla de espera y una pantalla principal, la cual es invocada a través de la función anidada *CreateTab* (Ver apartado 5.2.4).A continuación se ofrecen las primeras líneas de código donde se muestran algunos aspectos importantes que forman la interfaz.

En primer lugar se declaran todas las variables globales del programa. Se pueden destacar las variables EJE y S al ser variables de tipo estructura.

```
function main
 global ANCHO
 global ALTO
 global espera
 global interfaz
 global EJE
 global S
```

A continuación a través de la siguiente sentencia *get(0)* se pueden obtener todos los características de fábrica de MATLAB. Es nuestro caso, solo nos interesa los valores de las dimensiones de la pantalla, por lo que se concreta la sentencia con el parámetro *ScreenSize*. Como resultado, se muestra el siguiente array con los valores de la pantalla en la que se está ejecutando el programa [distancia desde la izquierda distancia desde el margen inferior ancho de la pantalla largo de la pantalla].

Las variables *ALTO* y *ANCHO* han sido declaradas globales porque serán empleadas más adelante para permitir que el tamaño de los elementos de la interfaz se ajuste al de la pantalla correctamente.

```
screensize = get(0,'ScreenSize');
    ANCHO = screensize (3);
    ALTO = screensize (4);
```

Debido a la cantidad de objetos gráficos que debe cargar MATLAB, se crea una pantalla de espera mientras que el programa carga toda la interfaz. Esta pantalla se hace a través de una figura *espera* definida con el tamaño de la pantalla anteriormente guardado y el color definido para todo el programa. Para indicar el mensaje de espera *'Loading interface...'* se debe incluir un uicontrol de tipo *"Text"*.

```
%%%%%%%%%%%----Figura de espera ----%%%%%%%%%%%%
    espera = figure(...
        'Name','',...
        'NumberTitle','off', ...
        'Menubar','none',...
        'Units','pixels',...
        'Position',[0,0,ANCHO,ALTO],...
        'Color',[0.078,0.169,0.549]);

    uicontrol(...
'Parent',espera,...
'Units','normalized','Position',[0.06,0.45,0.8,0.2],...
'Style','Text','BackgroundColor',[0.078,0.169,0.549],...
'Fontname','CourierNew','Fontsize',32,'Fontweight','Bold',...
'ForeGroundColor',[1 1 1],'FontAngle','normal',...
'String','Loading interface...');
```

La figura se maximiza mediante la función *maximize* [14] (Ver apartado 5.2.2.).

```
maximize(espera)
```

Cuando todos los objetos gráficos están cargados, se superpone una nueva figura interfaz.

Ciñéndonos al código, concretamente a las tres últimas líneas, la figura interfaz no aparecerá visible hasta que el programa pase por la función *CreateTab*, (Ver apartado 5.2.4) y llegue a la sentencia *set*. Seguidamente la figura se adapta a la pantalla cerrando la figura de espera.

```
interfaz = figure(...
        'Visible','off',...
        'Name','',...
        'NumberTitle','off', ...
        'Menubar','none',...
        'Units','normalized',...
        'Position',[0,0,1,1]);
```

```
function CreateTab(varargin)[…]


end
set(interfaz,'visible','On')
maximize(interfaz)
delete(espera)
end
```

5.2.4. Función CreateTab

Se trata de la función donde se declaran todos los objetos gráficos de la interfaz, al igual que contiene todas las subfunciones que se muestran en la figura 8.

A continuación se explica un ejemplo de los uicontrols más usados en esta función, para más detalle se recomienda ver el anexo 1 (apartado 11.1.5) donde se encuentra la programación completa.

Con la intención de que la interfaz quede ordenada se recurre a organizarlo mediante paneles y subpaneles. A continuación se muestran los paneles generales de entrada y salida, son paneles padres del resto de subpaneles.

```
E.panel_entrada=uipanel(...
        'Parent',hpanel(1),'ForeGroundColor',[1,1,1],...
        'Units','normalized','Position',[0.02 0.44 0.4 0.54],...
        'BackgroundColor',[0.82,0.863,0.922]);
E.panel_salida=uipanel(...
        'Parent',hpanel(1),'ForeGroundColor',[1,1,1],...
        'Units','normalized','Position',[0.02 0.02 0.25 0.4],...
        'BackgroundColor',[0.82,0.863,0.922]);
```

Para cargar cualquier imagen la programación utiliza la siguiente sintaxis. En este ejemplo se pinta el dibujo que contiene las fórmulas correspondientes a cada sistema.

Primero se declara el panel donde se quiere localizar.

```
E.panel_dibujos=uipanel(...
       'Parent',E.panel_entrada,'ForeGroundColor',[1,1,1],...
       'Units','normalized','Position',[0.03,0.65,0.94,0.28],...
       'BackgroundColor',[1 1 1]);
```

A continuación se destinan unos ejes para poder dibujarlo.

```
axes(...
     'Parent',E.panel_dibujos,...
     'Units','pixel',...
     'Position',[10,6,522,65]);
```

Finalmente mediante la sentencia *imread* e *imshow* se carga la imagen.

```
b=imread('dibujos.jpg');
imshow(b),axis off,hold on
```

Para finalizar, se menciona los dos tipos de uicontrol más utilizados a lo largo del programa.

En primer lugar se tiene un ejemplo de uicontrol que permite introducir texto.

```
E.dat_ganancia = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.75,0.60,0.23,0.20],...
'Style','edit','Fontname','CourierNew','Fontsize',10, ...
'Enable','Off',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');
```

Y también tenemos un ejemplo de uicontrol que solo te muestra texto.

```
E.texto_frecuencia=uicontrol(...
'Parent',E.subpanel_salida,'Units','normalized',...
'Position',[0.02,0.27,0.66,0.17],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','Output frequency (rad/s):',...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
```

.

5.2.5. Función CheckCallback

Para poder controlar el uicontrol de tipo *check* se crea esta función. A través del parámetro de entrada *hObject* se le pasa a la función el valor correspondiente de la acción realizada, en este caso el *check* marcado. Inicialmente ambos aparecen por defecto con un valor 0, es decir desmarcados.

```
%checkbox posición
    E.concatenar_1 = uicontrol(...
'Parent',E.panel_dibujos,'Units','Normalized',...
'Style','checkbox','Enable','On',...
'BackgroundColor',[1,1,1],...
'Position',[0.05,0.76,0.3,0.17],...
'Fontname','CourierNew','Fontsize',10,'String','PositionSystem',..
'Callback',@Check_Callback);

 %checkbox velocidad
     E.concatenar_2 = uicontrol(...
'Parent',E.panel_dibujos,'Units','Normalized',...
'Style','checkbox','Enable','On',...
'BackgroundColor',[1,1,1],...
'Position',[0.62,0.76,0.35,0.17],...
'Fontname','CourierNew','Fontsize',10,'String','VelocitySystem',..
'Callback',@Check_Callback);
```

Al marcar el *check E.concatenar_1* se cambia el valor a 1, entrando en el primer bucle *if*. Automáticamente por la lógica del bucle se activará una serie de uicontrols al igual que se pondrá el segundo *check E.concatenar_2* a 0. Uno de los uicontrols importantes que se habilita resulta ser el botón utilizado para ejecutar el programa, desde este se llamará a una nueva función *Postion_Data_Callback* (Ver apartado 5.2.6).

```
function Check_Callback(hObject, eventdata, handles)

 set( E.boton_new,'Enable','Off');

    if hObject == E.concatenar_1
      if get(E.concatenar_1,'Value') == 1


         set(E.concatenar_2,'value',0);
         set(E.dat_ganancia,'Enable','On');
         set(E.dat_cte.tiempo,'Enable','On');
         set(E.retardo,'Enable','On');
         set(E.amplitud_rele,'Enable','On');
         set(E.dat_frecuencia,'Enable','On');
         set(E.dat_rango,'Enable','On');
         set(E.retardo_1,'Enable','On');
         set(E.retardo_2,'Enable','On');


      %botón para activar el programa.
      E.boton_run= uicontrol (...
        'Parent',E.panel_entrada,'Units','Normalized',...
        'Position',[0.15,0.01,0.7,0.08],...
        'Style','pushbutton','Enable','On',...
        'String','Run',...
        'Fontname','Courier New','Fontsize',13,...
        'Callback',@Position_Data_Callback);
```

Para evitar que los uicontrols se queden activados si se desmarca el *check* se hace un *elseif* que inhabilita de nuevo los uicontrols.

```
       elseif get(E.concatenar_1,'Value') == 0


         set(E.dat_ganancia,'Enable','Off');
         set(E.dat_cte.tiempo,'Enable','Off');
         set(E.retardo,'Enable','Off');
         set(E.amplitud_rele,'Enable','Off');
         set(E.dat_frecuencia,'Enable','Off');
         set(E.dat_rango,'Enable','Off');
```

```
        set(E.retardo_1,'Enable','Off');
        set(E.retardo_2,'Enable','Off');
        set(E.boton_run,'Enable','Off');

    end
```

En el caso de que se pulse la otra opción disponible *E.concatenar_2* ocurrirá el mismo proceso pero en este caso desde el botón se llamará a una función diferente *Velocity_Data_Callback* (Ver apartado 5.2.7).

```
E.boton_run= uicontrol (...
       'Parent',E.panel_entrada,'Units','Normalized',...
       'Position',[0.15,0.01,0.7,0.08],...
       'Style','pushbutton','Enable','On',...
       'String','Run',...
       'Fontname','Courier New','Fontsize',13,...
       'Callback',@Velocity_Data_Callback);
```

5.2.6. Función Position_Data_Callback

Con objetivo de conectar nuestro programa principal con la función correspondiente al proceso elegido, en este caso para un controlador de posición, usamos esta función que es llamada desde la función *CheckCallback* (Ver apartado 5.2.5).

Primeramente se almacenan en unas nuevas variables de tipo estructura, los diferentes valores introducidos en los diferentes uicontrols pertenecientes al panel de entrada .La sentencia *get* nos facilita esta tarea.

```
function Position_Data_Callback(hObject, eventdata, handles)
```

```
E.string_ganancia=get(E.dat_ganancia,'string');
E.string_cte.tiempo=get( E.dat_cte.tiempo,'string');
E.string_retardo=get(E.retardo,'string');
E.string_frecuencia=get(E.dat_frecuencia,'string');
E.string_rango=get(E.dat_rango,'string');
E.string_amplitud_rele=get( E.amplitud_rele,'string');
E.string_retardo_1=get( E.retardo_1 ,'string');
E.string_retardo_2=get( E.retardo_2 ,'string');
```

Para evitar que alguno de los campos se quede vacío, se hace un bucle donde si alguna de las anteriores variables aparece vacía automáticamente a través de la sentencia *msgbox* se despliega un mensaje de error.

```
if strcmp('',E.string_ganancia) ||
strcmp('',E.string_cte.tiempo)||strcmp('',E.string_retardo)||...
strcmp('',E.string_rango)||strcmp('', E.string_frecuencia)|| ...
strcmp('',E.string_amplitud_rele)||strcmp('',E.string_retardo_1||.
..strcmp('',E.string_retardo_2)
msgbox('Input data missing.Please check','Error','error');
```

Debido a que los datos de los uicontrols son de tipo carácter, antes de llamar a la función que realiza todas las operaciones, se deben transformar los datos a tipo entero. Esto se hace a través de la sentencia *str2num.*

```
  else
S.Gain = str2num(E.string_ganancia);
S.Time_constant = str2num(E.string_cte.tiempo);
S.Delay = str2num(E.string_retardo);
S.User_frequency = str2num(E.string_frecuencia);
S.Frequency_range = str2num(E.string_rango);
S.Relay_amplitude =str2num(E.string_amplitud_rele);
S.First_delay = str2num(E.string_retardo_1);
S.Second_delay = str2num(E.string_retardo_2);
```

A continuación se llega a una de las partes más importantes del programa, la llamada a la función externa Rele_Param_Delay_Pos (Ver apartado 3.1). Como

datos de entrada se le pasan los anteriormente transformados, y como datos de salida la función devuelve valores que utilizaremos más tarde.

```
[S.Output_amplitude,S.Output_period,S.Output_frequency,S.Final_del
ay,S.Magnitude_estimated,...
S.Magnitude_real,S.Phase_estimated,S.Phase_real]=Rele_Param_Delay_
Pos(S.Gain,S.Time_constant,S.Delay,...
S.User_frequency,S.Frequency_range,S.Relay_amplitude,S.First_delay
,S.Second_delay,...
EJE.grafica_iteracion,EJE.grafica_bode_magnitud,EJE.grafica_bode_f
ase);
```

Se observa que los cuatro últimos valores de entradas no están definidos dentro de esta función, pero sí dentro de la función *CreateTab*. Al pertenecer a una estructura *EJE* declarada como global al principio del programa, los valores se podrán utilizar sin problemas dentro del programa.

Una vez realizadas todas las operaciones y recibido sin problemas los datos de salida, al tratarse de datos de tipo entero y querer mostrarse sobre los uicontrols de salida se debe hacer el proceso inverso, pasar los valores de tipo entero a string. En este caso se usa la sentencia *num2str* donde se le especifica una precisión que el valor solo tiene que tener cuatro letras. Seguidamente se llama a la función *Display_Data* (Ver apartado 3.2)

```
E.conv_amplitud= num2str(S.Output_amplitude,4);
E.conv_periodo= num2str(S.Output_period,4);
E.conv_frecuencia_1= num2str(S.Output_frequency,4);
E.conv_delay= num2str(S.Final_delay,4);
E.conv_magnitud= num2str(S.Magnitude_estimated,4);
E.conv_magnitud_1= num2str(S.Magnitude_real,4);
E.conv_fase= num2str(S.Phase_estimated,4);
E.conv_fase_1= num2str(S.Phase_real,4);
```

```
%llamada a una subfunción
  Display_Data ;
 end
end
```

5.2.7. Función Velocity_Data_Callback

Al tratarse de una función similar a la anteriormente expuesta (Ver apartado 5.2.6), se detallan solo las diferencias existentes.

Básicamente la función *Velocity_Data_Callback* se encarga de recibir los datos del panel de entrada, comparar que ninguno dato este vacío y finalmente llamar a la función externa correspondiente al proceso del controlador de velocidad.

```
function Velocity_Data_Callback(hObject, eventdata, handles)

E.vel_ganancia=get(E.dat_ganancia,'string');
E.vel_cte.tiempo=get( E.dat_cte.tiempo,'string');
E.vel_retardo=get(E.retardo,'string');
E.vel_frecuencia=get(E.dat_frecuencia,'string');
E.vel_rango=get(E.dat_rango,'string');
E.vel_amplitud_rele=get( E.amplitud_rele,'string');
E.vel_retardo_1=get( E.retardo_1 ,'string');
E.vel_retardo_2=get( E.retardo_2 ,'string');

%compara que no haya ninguna caja vacía
if strcmp('',E.string_ganancia) ||
strcmp('',E.string_cte.tiempo)|| strcmp('',E.string_retardo)||...
        strcmp('',E.string_rango)||strcmp('',
E.string_frecuencia)|| strcmp('',
E.string_amplitud_rele)||strcmp('',E.string_retardo_1)||
strcmp('',E.string_retardo_2)
msgbox('Input data missing.Please check','Error','error');

  else
```

```
E.num_vel_ganancia = str2num(E.vel_ganancia);
E.num_vel_cte.tiempo = str2num(E.vel_cte.tiempo);
E.num_vel_retardo = str2num(E.vel_retardo);
E.num_vel_frecuencia = str2num(E.vel_frecuencia);
E.num_vel_rango = str2num(E.vel_rango);
E.num_vel_am_rele =str2num(E.vel_amplitud_rele);
E.num_vel_retardo_1 = str2num(E.vel_retardo_1);
E.num_vel_retardo_2 = str2num(E.vel_retardo_2);
```

En este caso la función externa a la que se llama es *Rele_Param_Delay*, los datos de entrada aparecen marcados en azul y los de salida en rojo.

```
[S.Output_amplitude,S.Output_period,S.Output_frequency,S.Final_del
ay,S.Magnitude_estimated,...
S.Magnitude_real,S.Phase_estimated,S.Phase_real]=Rele_Param_Delay(
E.num_vel_ganancia,E.num_vel_cte.tiempo,E.num_vel_retardo,...

E.num_vel_frecuencia,E.num_vel_rango,E.num_vel_am_rele,E.num_vel_r
etardo_1 ,E.num_vel_retardo_2,...

EJE.grafica_iteracion,EJE.grafica_bode_magnitud,EJE.grafica_bode_f
ase);


%conversion de variables que vienen de la funcion de num a string
   E.conv_amplitud= num2str(S.Output_amplitude,4);
   E.conv_periodo= num2str(S.Output_period,4);
   E.conv_frecuencia_1= num2str(S.Output_frequency,4);
   E.conv_delay= num2str(S.Final_delay,4);
   E.conv_magnitud= num2str(S.Magnitude_estimated,4);
   E.conv_magnitud_1= num2str(S.Magnitude_real,4);
   E.conv_fase= num2str(S.Phase_estimated,4);
   E.conv_fase_1= num2str(S.Phase_real,4);


  Display_Data;

end
end
```

5.2.8. Función Display_Data

Se trata de una función sencilla encargada de la visualización de los resultados. Simplemente pinta de nuevo los uicontrols de tipo *text*, ya definidos en la función *CreateTab*(ver punto 5.2.4) ,con la salvedad, de que ahora contienen una nueva propiedad *string* que es la encargada de pintar la correspondiente variable de salida.

```
function Display_Data ()
```

Al tratarse de la misma sintaxis se muestra solo un uicontrol.

```
E.sal_amplitud=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.76,0.21,0.20],...
'Style','text','Fontname','CourierNew','Fontsize',10,...
'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string',E.conv_amplitud);
```

En la figura 12 se muestra una imagen que relaciona cada uicontrol de esta función con su correspondiente cuadro de salida en la interfaz.

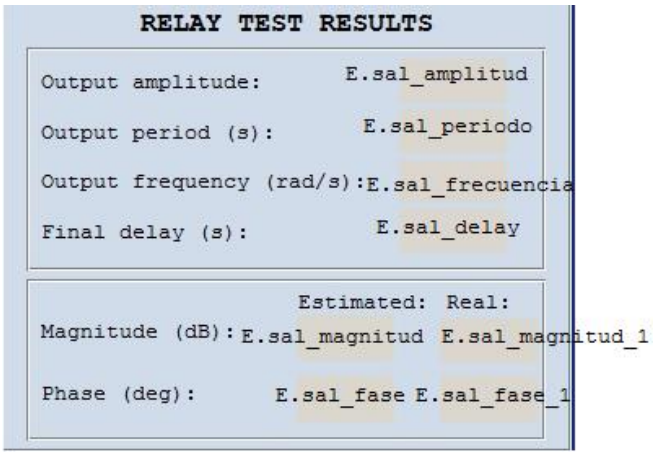


Figura 12: Panel de salida con sus uicontrols

En el siguiente paso se activará el botón *New Data* para permitir al usuario realizar un nuevo proceso. Al activarse este, se desactivan el resto de botones hasta recibir la llamada al nuevo proceso.

```
      E.boton_new= uicontrol (…
          'Parent',E.panel_entrada,'Units','Normalized',…
          'Position',[0.05,0.01,0.2,0.08],…
          'Style','pushbutton','Enable','On',…
          'String','New Data',…
          'Fontname','Courier New','Fontsize',12,…
     'callback',@New_data);
        set(E.boton_run,'enable','off');
        set(E.concatenar_1,'enable','off');
        set(E.concatenar_2,'enable','off');

end
```

5.2.9. Función New_data

Se trata de una función encargada de borrar todos los datos de la interfaz. Para realizar esta acción se juega con la sentencia *set* para los uicontrol de tipo *edit* y *text.*

```
function New_data(hObject, eventdata, handles)

          set(E.concatenar_1,'enable','on');
          set(E.concatenar_2,'enable','on');
          set(E.dat_ganancia,'String','','enable','off');
```

Y para el caso de las gráficas se pintan de nuevo vacías.

5.2.10. Función Save_Data

Función encargada de guardar todos los datos pertenecientes a la estructura S en un archivo de tipo mat .El destino donde se guardan los datos es elegido por el usuario.

```
function Save_Data(hObject, eventdata, handles)
  [filename, pathname] = uiputfile('*.mat');
  save(filename,'*S')
    clear *S;
    end
```

Para acceder a esta función se crea el siguiente uicontrol:

```
E.boton_save= uicontrol (...
'Parent',hpanel(1),'Units','Normalized',...
'Position',[0.28,0.05,0.15,0.05],...
'Style','pushbutton','Enable','On',...
'String','Save experiment data(.mat)',...
'Fontname','CourierNew','Fontsize',10,'callback',@Save_Data);
```

5.2.11. Función Save_Interface

En esta caso, se usa de nuevo otro uicontrol de tipo *pushbutton* que llama al hacer click llama a la función Save_Interface.

```
E.boton_save_screen= uicontrol (...
'Parent',hpanel(1),'Units','Normalized',...
'Position',[0.28,0.12,0.15,0.05],...
'Style','pushbutton','Enable','On',...
'String','Save screenshot',...
'Fontname','CourierNew','Fontsize',10,'callback',@Save_Interface);
```

Esta función te permite exportar mediante la sentencia *hgexport* una imagen de la figura que se está visualizando en el momento de pulsar el botón. De nuevo el destino donde se guarda la imagen es elegido por el usuario.

```
function Save_Interface(varargin)
[filename, pathname] = uiputfile('*.jpg');
      hgexport(gcf, filename,hgexport('factorystyle'), 'Format',
'jpeg');
    end
end
```

5.2.12. Función Read_PDF

Al hacer click sobre el uicontrol *E.boton_ayuda* se llama a la función Read_PDF.

```
E.boton_ayuda= uicontrol (...
'Parent',hpanel(1),'Units','Normalized',...
'Position',[0.31,0.22,0.07,0.07],...
'Style','pushbutton','Enable','On',...
'String','HELP',...
'Fontname','Courier New','Fontsize',11,'callback',@Read_PDF);
```

Esta función a través de una sentencia propia de Matlab, abre el archivo que se le indica entre paréntesis, en este caso es un documento que servirá de ayuda al usuario. Cabe mencionar que esta sentencia solo abre archivos de tipo pdf, txt,…

```
function Read_PDF (varargin)
open('Help.pdf');
end
```

6. FUNCIONAMIENTO DE LA INTERFAZ

6.1. Requisitos para su funcionamiento

Para poder hacer uso de la interfaz es imprescindible que el usuario tenga instalado el programa MATLAB en su equipo. Desde la pantalla principal de MATLAB debe cargarse en el Current Folder la carpeta donde se encuentra localizados todos los archivos referentes al programa. A continuación basta con abrir el archivo principal *main.m* y darle al play.

Plataformas compatibles

La interfaz ha sido programada en la versión R2012a de Matlab compatible con todos los sistemas operativos que contenga las versiones posteriores a la del 2012 incluida .Si se usa versiones anteriores es conveniente asegurarse que estas tienen todas las librerías necesarias.

Una vez creada toda la interfaz a partir de la programación explicada en los apartados 3 y 5, se pasa a detallar el funcionamiento de la misma. Para un buen uso de la interfaz se recomienda leer todos los pasos detenidamente además de tener unos mínimos conocimientos del test de relé.

6.2. Funcionamiento paso a paso

Antes de empezar el apartado debe aclararse que para un mejor entendimiento del funcionamiento de la interfaz se utiliza como ejemplo un sistema de posición.

Debido a la cantidad de archivos que la interfaz tiene que procesar tras inicializar el programa, se muestra una primera pantalla de espera la cual contiene su correspondiente mensaje para indicar que la interfaz se está cargando (Ver figura 13).

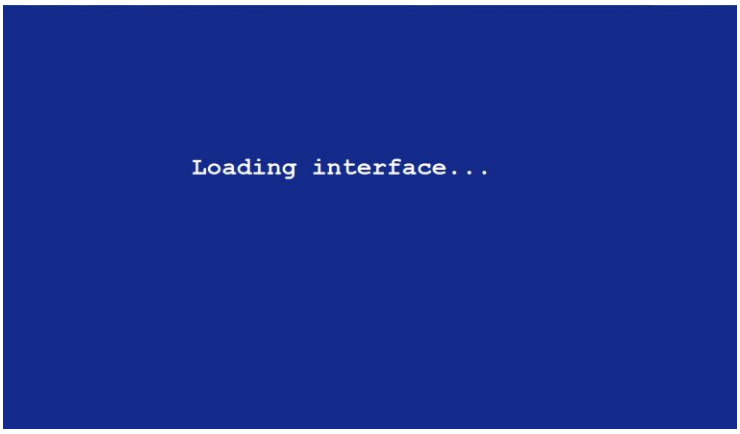


Figura 13: Pantalla de espera.

Pasados unos instantes, el programa comienza a funcionar mostrando dentro de una pestaña la pantalla principal. Aquí, se encuentran localizados los diferentes elementos necesarios para que el usuario pueda realizar el método del test de relé modificado sin dificultades (Ver figura 14).

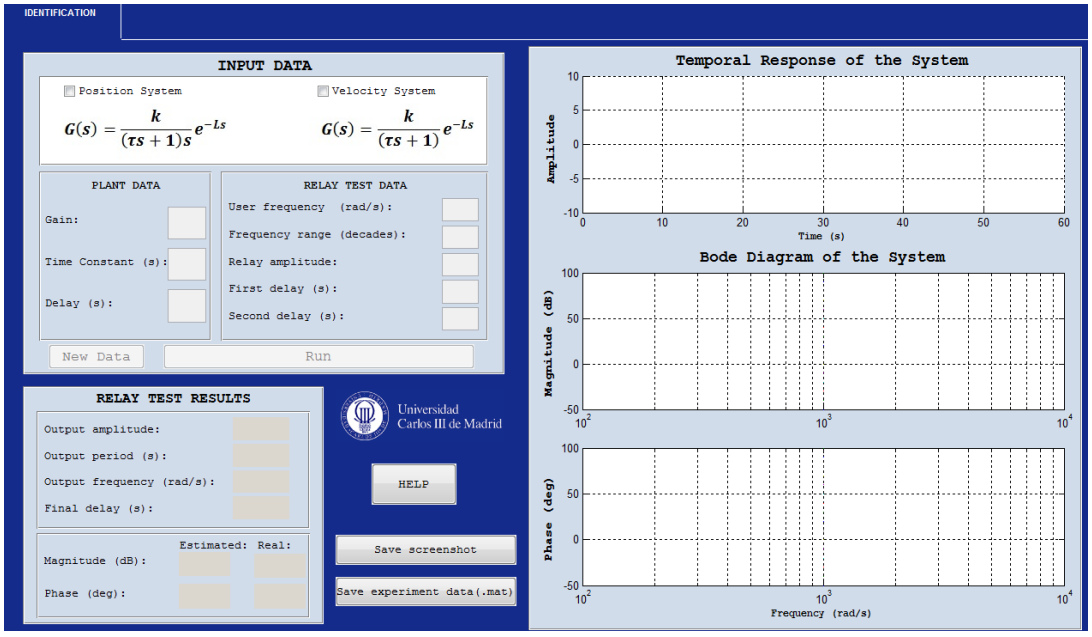


Figura 14: Pantalla principal del programa.

El primer paso se realiza sobre el panel de entrada (input data). donde se debe elegir el tipo de sistema dinámico que se quiere identificar. El método test de relé modificado dispone de dos opciones sistema de posición o de velocidad (Position System o Velocity System). Debido a que la interfaz está programada para que no se puedan simular diferentes tipos de sistemas a la vez, es imprescindible elegir uno de ellos para que se habiliten los datos de la planta (plant data), del test de relé (Relay test data) y el botón “Run”.).En este ejemplo se opta por la primera opción y se continua con el proceso.

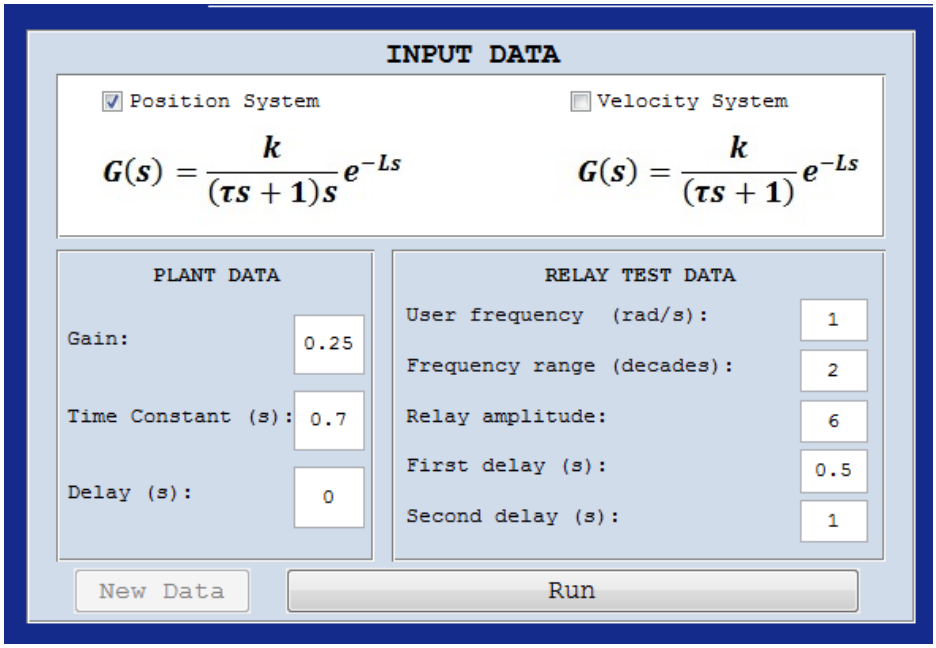


Figura 15: Panel de datos de entrada.

Tras seleccionar el sistema deseado, se deben introducir por un lado los datos encargados de definir el sistema que se quiere identificar. En nuestro ejemplo se define el sistema con una ganancia (Gain) de 0.25, una constante de tiempo (Time constante) de 0.7 segundo y sin retardo (Delay) .

Por otro lado se introducen los datos necesarios para el método del test de relé(Relay test data):

- Frecuencia del usuario deseada (User frequency) →1 radian/segundo.
- Rango de frecuencia (Frequency range) → 2 décadas.
- Amplitud del relé (Relay amplitude) → 6.
- Primer retardo (First delay) → 0.5 segundos.
- Segundo retardo (Second delay) → 1 segundo.

Cuando tenemos todo el panel completo, se pulsa sobre el botón “Run”, si falta algún campo por rellenar, el programa muestra una ventana con un mensaje de error (Ver figura 16).

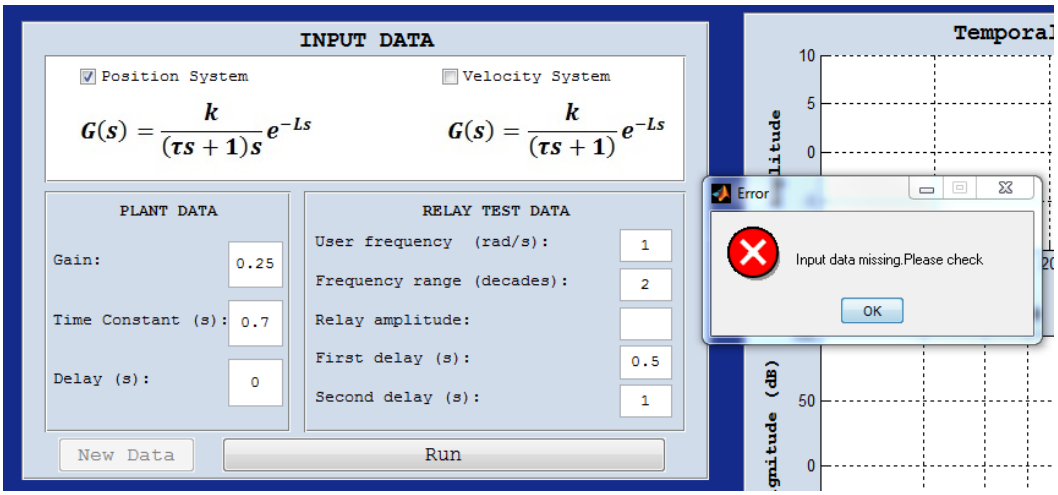


Figura 16: Ventana de error por falta de datos.

En caso contrario, se empieza con el proceso. Para indicar al usuario que el proceso tarda unos segundos, se visualiza la siguiente pantalla de espera (Ver Figura 17).

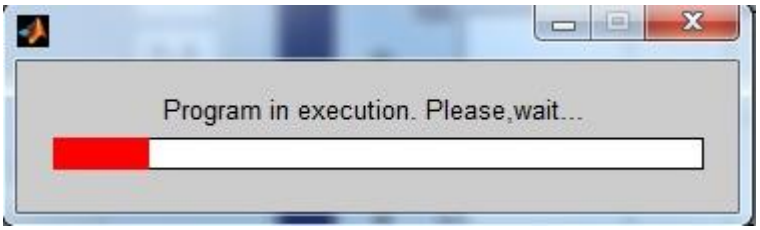


Figura 17: Ventana de espera

Tras realizarse el proceso con éxito, se pasa a la segunda parte de la pantalla, los resultados del test de relé (relé test results).

Para interpretar mejor los resultados, el panel se divide en dos recuadros. En el primero se obtienen los datos de salida correspondientes a la respuesta temporal del sistema. Como resultados del ejemplo tenemos:

- Amplitud de salida (Output amplitude): 1.83. Es el valor perteneciente a la amplitud de la onda.
- Periodo de salida (Output period): 6.8 segundos. Este valor proviene del periodo de la onda
- Frecuencia de salida (Output frequency): 0.924 radianes/segundos. Este valor corresponde a la frecuencia en la que se para el proceso, en el resultado es importante comprobar si la diferencia entra la frecuencia de entrada y de salida es menor de 0.15radianes/segundos, para dar validez al mismo.
- Retardo final (Final delay):1 segundo. Se trata último valor del retardo que utiliza el sistema para alcanzar la frecuencia deseada.

Si pasamos al segundo recuadro, se obtienen los datos de magnitud y fase obtenidos a partir de los resultados del primer recuadro.
Se observa que para ambas datos se tienen dos valores diferentes. El valor estimado es el resultado tras utilizar el método del test de relé modificado, en cambio el valor real es el resultado obtenido tras hacer el cálculo teórico.

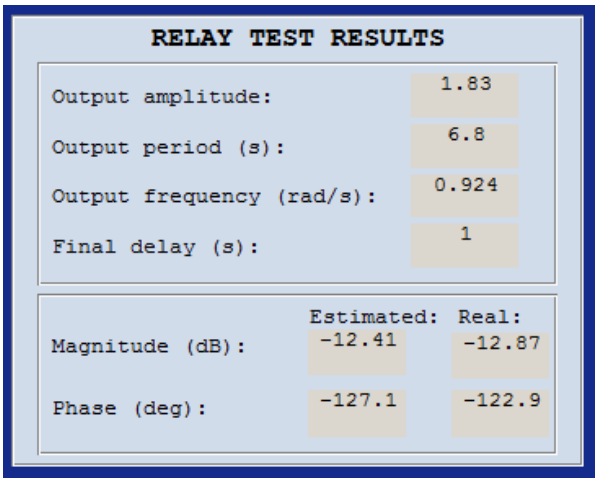


Figura 18: Panel datos de salida

Con intención de facilitar al usuario la interpretación de los resultados obtenidos, sobre un tercer panel situado a la derecha, el programa la respuesta temporal de la última iteración del sistema (Ver figura 19).

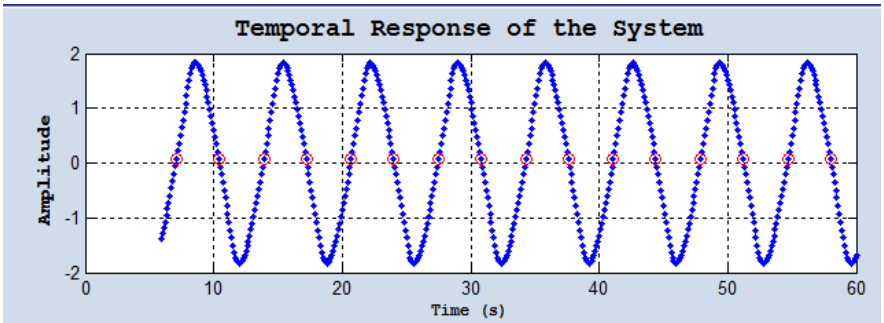


Figura 19: Respuesta temporal del sistema

A continuación se representa el diagrama de Bode correspondiente al sistema real definido por el usuario. Según este diagrama, y tal y como arroja la interfaz, la magnitud y fase a la frecuencia del usuario, son -12.8 (dB) y -122.9 (grados) respectivamente (Ver figura 20).Mientras que los calores estimados por el test del relé son -12.41 (dB) y -121.1 (grados).

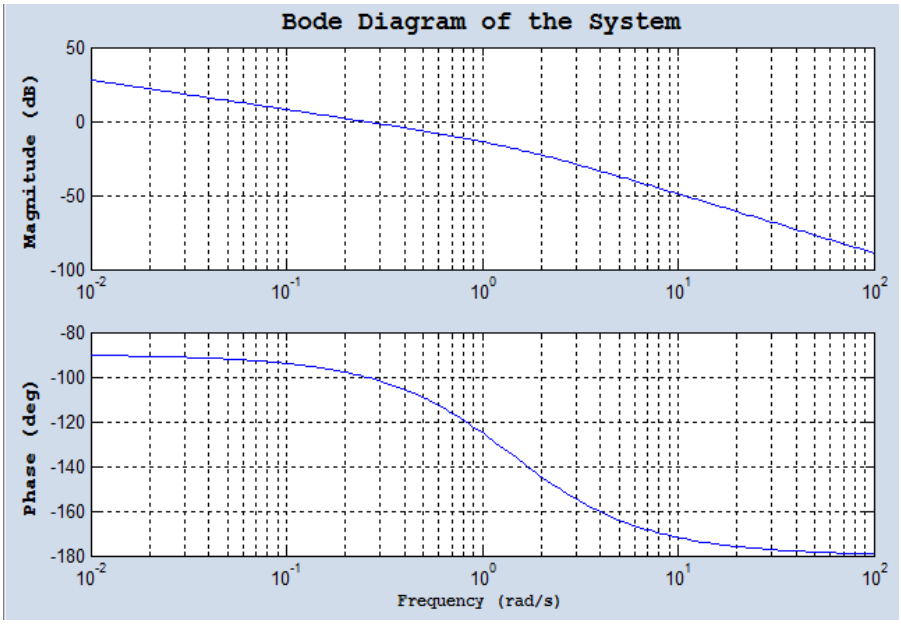


Figura 20: Diagrama de Bode

Para permitir al usuario entrar en detalle y reutilizar todas las gráficas del proceso, el programa las guarda automáticamente en un archivo denominado ***gráficas.fig*** (Ver figura 21).

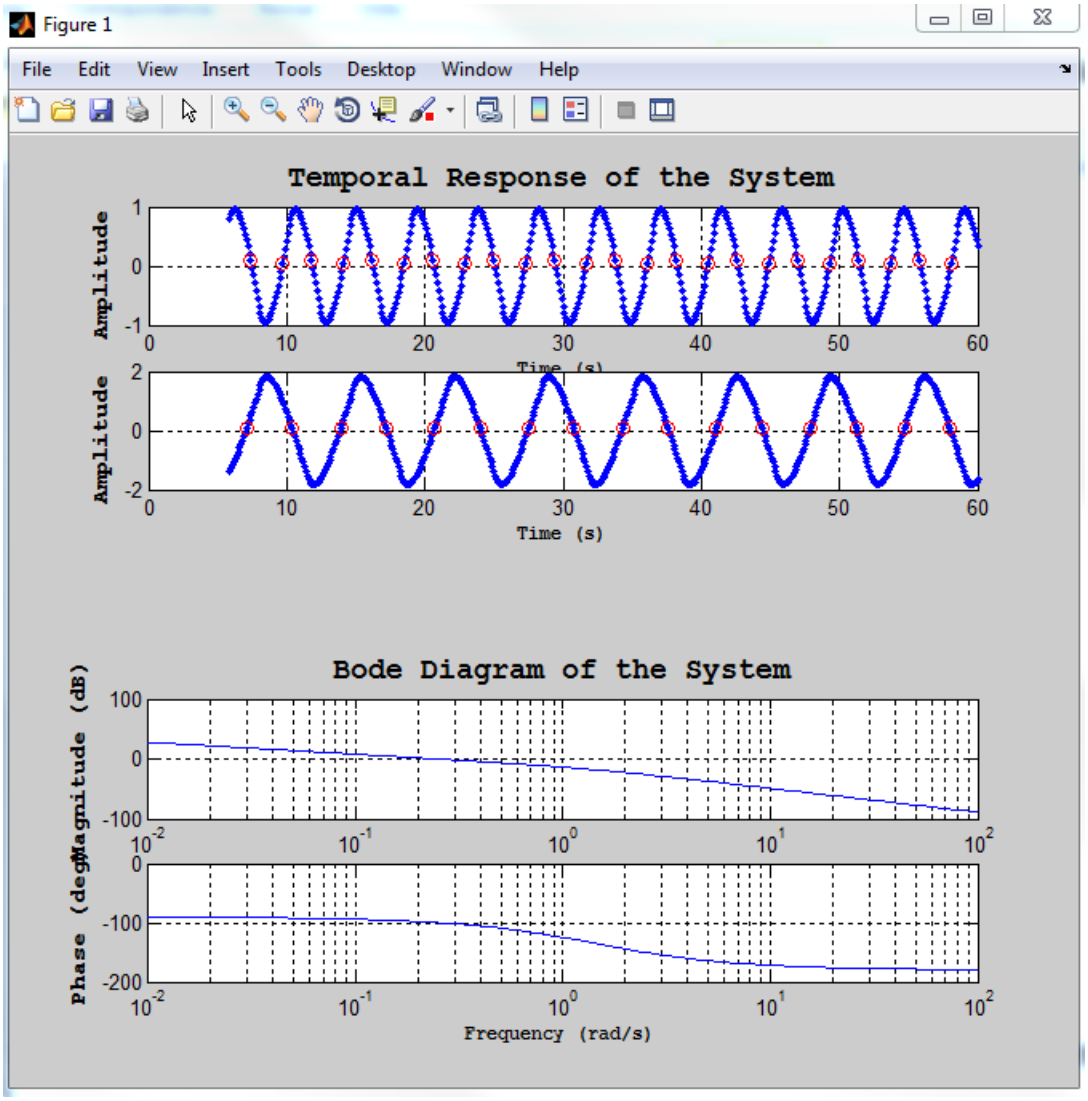


Figura 21: Archivo *gráficas.fig*

Opcionalmente, la interfaz también ofrece la posibilidad de guardar una imagen de toda la pantalla a través del botón "Save screenshot".

También con el botón "Save experimet data (.mat)" se almacenan los datos de entrada y de salida en un fichero de tipo mat.

En ambos casos el usuario elige el destino de estos archivos.

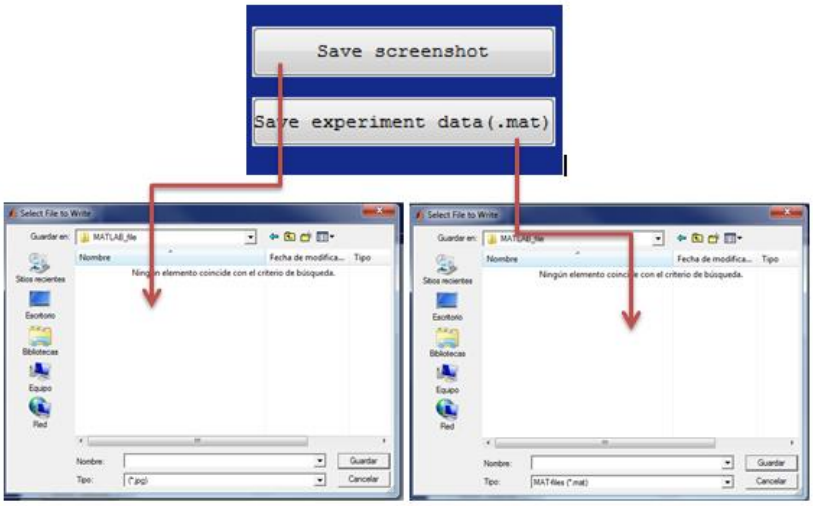


Figura 22 : Botones *Save screenshot* y *Save experiment data (.mat)*

Con la intención de evitar que el usuario pueda tener algunas dudas acerca de los pasos a realizar sobre la interfaz, existe el botón de ayuda "HELP". Al pulsar sobre él, automáticamente se abre un documento con formato PDF con las instrucciones.

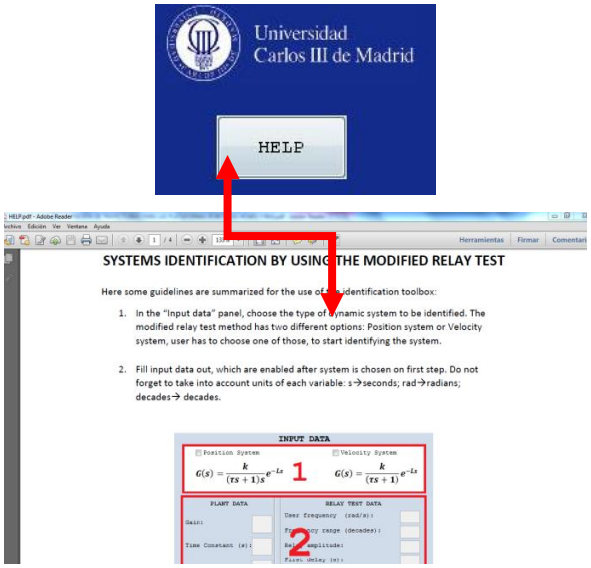


Figura 23: Botón *HELP*

7. MARCO SOCIO-ECONÓMICO

7.1. Planificación

El desarrollo de este trabajo comenzó el 9 de Septiembre del 2013. Inicialmente se planificaron las distintas etapas del proyecto, que se han dividido a lo largo del desarrollo del mismo en las indicadas a continuación.

En un primer bloque se realizó el estudio de la herramienta GUIDE. A partir de otros proyectos realizados anteriormente y de búsquedas en internet, conseguimos conocer los conceptos básicos de esta herramienta para posteriormente usarla.

En un segundo bloque se realizó el estudio del método del test de relé a implementar. En esta fase se adquirieron todos los conocimientos sobre el método y sus posibles alternativas de implementación.

En un tercer bloque, y podríamos decir que el más importante, nos centramos en la implementación del test del relé modificado en Matlab, a través de su herramienta GUIDE. Resulta ser la fase más compleja porque es la encargada de cumplir los objetivos.

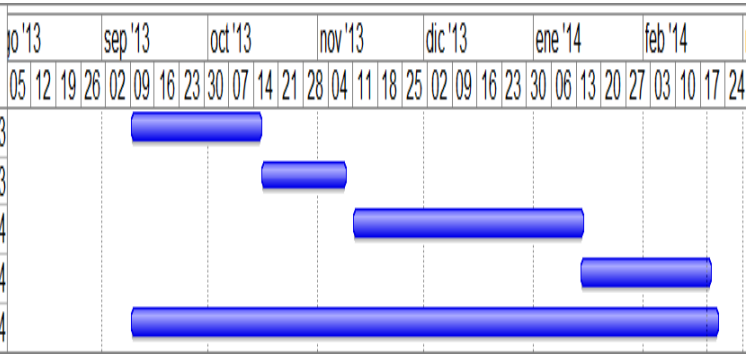
En un cuarto bloque donde se analizan los resultados obtenidos tras finalizar el trabajo y se hacen las pruebas de testeo del test. Esta fase se considera de gran importancia porque permitirá corregir los posibles errores que se puedan haber detectado anteriormente.

Finalmente podemos incluir un bloque extra, donde se realiza la memoria del trabajo.

A continuación se puede ver el diagrama del proyecto donde se estima el tiempo empleado en cada una de las fases.

Tabla 1: Diagrama de Gantt

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Estudio de la herramienta GUIDE	27 días	lun 09/09/13	mar 15/10/13
2	Estudio del método test de relé modificado y sus alternativas	18 días	mié 16/10/13	vie 08/11/13
3	Diseño de la interfaz	47 días	lun 11/11/13	mar 14/01/14
4	Pruebas y análisis de los resultados	27 días	mar 14/01/14	mié 19/02/14
5	Elaboración de la memoria del trabajo	120 días	lun 09/09/13	vie 21/02/14



7.2. Presupuesto

A continuación se hace un presupuesto de los costes que tiene la elaboración de este trabajo fin de grado, tanto a nivel personal como de costes de software y hardware [9].

Coste Personal			
Categoría	**Nº Horas**	**Coste hora**	**Coste Total**
Jefe de proyecto	48	35 €	1680 €
Ingeniero Junior	360	10 €	3600 €
		TOTAL	**5280 €**

Tabla 2: Coste Personal

Para calcular los costes tanto de software como de hardware se hace uso de la siguiente fórmula:

Costes= (A/B) x C x D

A= Número de meses desde la fecha de facturación en que el equipo es utilizado.

B=Periodo de depreciación.

C=Coste del equipo (sin IVA)

D=Porcentaje de uso que se dedica al proyecto. Se fija al 100%.

Descripción	Coste	Dedicación(meses)	Periodo de depreciación (meses)	Coste Imputable
Microsoft Windows 7 Professional	130,2 €	6	60	13,02 €
Microsoft Office 2010	0 €	6	60	0 €
Matlab R2011a	2.100 €	6	60	210 €
Adobe Reader	0 €	6	60	0 €
Costes Hardware				
Portátil Toshiba Satélite C55-A-1NV	649 €	6	60	64,90 €
			TOTAL	287,92 €

Tabla 3: Costes de Software y Hardware

Costes totales	
CONCEPTO	Coste Imputable
Coste Personal	5280 €
Costes Software	223,02 €
Costes Hardware	64,9 €
Beneficios Empresariales (30%)	1670,38 €
TOTAL	7238,32 €

Tabla 4: Costes totales

8. CONCLUSIONES

Llegados a este apartado, es necesario recordar los objetivos que en un principio se plantearon y analizar el cumplimiento de ellos. Como objetivo principal se pretendía la implementación del método del test de relé modificado sobre una interfaz gráfica a través de la herramienta GUIDE. Después de la elaboración de todo el trabajo, el objetivo puede darse por superado, aunque se deben destacar algunas dificultades encontradas en el camino.

La primera dificultad y sin lugar a dudas la más importante debido a que se trataba del punto fuerte del trabajo, fue la conexión entre la interfaz y el test de relé modificado. Hubo bastantes problemas con la transferencia de datos, pero finalmente se llegó a la solución desarrollada en esta memoria, aunque se recomienda mejorarla en un futuro.

Seguidamente encontramos problemas a la hora de guardar las gráficas mediante un botón de la interfaz. Cómo alternativa para cumplir igualmente este requisito, se optó por guardar una imagen de toda la pantalla que muestra la interfaz, opción a la que sí que se puede acceder a través del botón “Screenshot” de la interfaz.

Pero debido a la importancia de poder trabajar con estas gráficas en un futuro, se pensó igualmente guardarlas automáticamente según se generase el proceso en un archivo aparte.

Otra de las dificultades que podemos citar es acerca del documento de ayuda. En un primer momento se pensó incorporar el PDF dentro de la propia interfaz como un elemento más. Debido a las limitaciones que esta herramienta tiene para pintar archivos con formato texto, se eligió la alternativa de abrirlo a través del botón “HELP”.

Al igual que se hace una análisis crítico del trabajo experimental considero importante hacerlo a nivel personal. A lo largo de todo el trabajo destacaría la capacidad para resolver problemas que se desarrolla y la cantidad de conocimientos nuevos que se adquieren.

Durante toda la carrera es verdad que te enfrentas a problemas muy complicados, pero la diferencia principal con respecto a un trabajo fin de grado es que siempre se cuenta con el respaldo de compañeros y profesores. Un trabajo fin de grado es un trabajo solo y exclusivo de una persona, de ahí surge la gran satisfacción cuando todo sale correcto o la frustración cuando por el contrario no encuentras la solución. A lo largo de todo este trabajo puedo afirmar que he pasado por esos estados y por muchos más típicos de un trabajo de este nivel.

9. TRABAJOS FUTUROS

Aunque a simple vista hemos concluido que el trabajo podría estar listo para su uso, se proponen algunas mejoras inspiradas en las dificultades encontradas a lo largo del mismo, al igual que se proponen nuevas ideas para hacer el trabajo más completo.

Mejoras propuestas:

- Depurar el código de la interfaz, es decir, buscar alternativas para todo el proceso de conexión.
- Guardar las gráficas directamente a través de un botón de la interfaz.
- Añadir una nueva pestaña donde se incluya el documento de ayuda como un elemento más de la interfaz.

Ideas nuevas:

- Aplicar el método del test de relé modificado a una plataforma real. Tras comprobar con el proceso de simulación la fiabilidad del método de identificación test de relé, se propone cambiar el archivo simulink por una plataforma real cuyo modelo se desconoce. Esta mejora conllevaría hacer ciertos cambios en la programación de la interfaz. Actualmente es un trabajo que se encuentra en proceso.

- Incorporar otros métodos de identificación sobre la misma interfaz. Para ello se podrían usar diferentes pestañas, opción que se ha dejado facilitada en el código.

10. REFERENCIAS

MANUALES

[1]Carrera Amuriza, A.R., Martínez Nebreda, M. "Introducción a MATLAB y a la creación de interfaces gráficas." Servicio Editorial de la Universidad del País Vasco. 2004.

[2] "MATLAB 7 Getting Started Guide." Mathworks. 2010.

[3] Isermann, Rolf, M, M. "Identification of Dynamic Systems", 2011.

[4] Royo, Javier. "Diseño Digital". Ediciones Paidós Ibérica, 2004.

[5] Shneiderman, Ben." Designing the user interface, Strategies for effective Human-computer interaction". Addison-wesley. 1998.

PROYECTOS FIN DE CARRERA

[6] Kunush, Cristian "IDENTIFICACIÓN DE SISTEMAS DINÁMICOS". Universidad Nacional de la Plata. Facultad de Ingeniería. 2003.

[7] Sabio Gallego, F.B "INTERFAZ PARA LA GENERACIÓN DE TRAYECTORIAS PARA LAS PLATAFORMAS ROBÓTICAS HOAP-3 Y RH-2". Universidad Carlos III de Madrid. 2011.

[8] Carrillo Valencia, V. "AUTOSINTONÍA DE CONTROLADORES PID FRACCIONARIOS MEDIANTE UN AUTÓMATA PROGRAMABLE". Universidad Carlos III de Madrid. 2009.

[9] Leal Díazz, Oscar. "OBTENCÍON, TRATAMIENTO Y VISUALIZACIÓN DE ESTADÍSTICAS MEDIANTE HERRAMIENTA WEB".Universidad Carlos II de Madrid.2012.

DIRECCIONES DE INTERNET

[10] Jack Little, C. M. (1994-2014). MathWorks. Recuperado el 1 de 02 de 2014, de http://www.mathworks.es/es/help/matlab/ref/subplot.htm

[11] Jack Little, C. M. (1994-2014). MathWorks. Recuperado el 5 de 11 de 2013, de http://www.mathworks.es/products/matlab/

[12] Jack Little, C. M. (1994-2014). MathWorks. Recuperado el 9 de 01 de 2014, de http://www.mathworks.es/mobile/index.html

[13] Jack Little, C. M. (1994-2014). MathWorks. Recuperado el 9 de 01 de 2014, de http://www.mathworks.es/es/help/matlab/creating_guis/customizing-callbacks-in-guide.html

[14] Archivo maximize.m . Jack Little, C. M. (1994-2014). MathWorks. Recuperado el 28 de 03 de 2010, de http://www.mathworks.se/matlabcentral/fileexchange/25471-maximize

[15] Archivo uitabpanel.m . Jack Little, C. M. (1994-2014). MathWorks. Recuperado el 28 de 03 de 2010, de http://www.mathworks.com/matlabcentral/fileexchange/11546

11. ANEXOS

11.1. ANEXO 1: CÓDIGO PROGRAMADO

En esta sección se ofrece el código completo del trabajo fin de grado, para que el lector pueda consultar con más detalle los apartados 3 y 5.

11.1.1. Test de relé posición

Se detalla toda la programación referente al test del relé con un sistema de posición.

```
function[amplitud,periodo_error,wu,L_rele,magnitud_estimada,magnit
ud_real,arg_estimado,arg_real]=
Rele_Param_Delay_Pos(k,tau,L,wc,rango,d,...
L_rele_1,L_rele_2,g_iteracion,grafica_1,grafica_2)

w_total=0;

L_total=0;

L_rele=L_rele_1;
assignin('base','L_rele', L_rele
%wc=frecuencia de corte deseada
assignin('base', 'wc', wc);
%k=ganancia
assignin('base','k', k);
%tau=cte. de tiempo;
assignin('base','tau', tau);
%L=retardo planta;
assignin('base','L', L);
%d=amplitud del rele;
assignin('base', 'd', d);
%rango=rango de frecuencias;
assignin('base', 'rango', rango);

%Barra de espera
bp=waitbar(0,'Program in execution. Please,wait...','Name','');

M=40;
I=0;

while I<M

I=I+1;
H=2*I;
G=I+3*I;
waitbar(I/M,bp);
sim('rele_delay_posicion')
```

```
end

delete(bp)

relesinref=[rele(:,1),rele(:,2)];

t=relesinref(60:length(relesinref),1);
error=relesinref(60:length(relesinref),2);

amplitud_max=max(error)

pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
if sign(error(ind)) ~= sign(anterior)

if cont==0
error1=error(ind:length(error));
amplitud_min=min(error1)
amplitud=(amplitud_max-amplitud_min)/2
cont=1;
end

if error(ind) > 0
pos=[pos ind];
else
pos=[pos ind-1];
end
end
anterior=error(ind);
end
figura=figure;
subplot(5,1,1)
plot(t,error,'.',t(pos),error(pos),'or'),grid on;
title ('Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
xlabel
('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
grid on;

plot(g_iteracion,t,error,'.',t(pos),error(pos),'or'),grid on;
title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,'Fontna
me','Courier New');
xlabel
```

```
(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,'Fontname',
'Courier New');
grid (g_iteracion);

pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error

w_total=[w_total wu]

L_total=[L_total L_rele]

L_rele=L_rele_2;
assignin('base', 'L_rele', L_rele);
sim('rele_delay_posicion')
relesinref=[rele(:,1),rele(:,2)];

t=relesinref(60:length(relesinref),1);
error=relesinref(60:length(relesinref),2);
amplitud_max=max(error)

pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
if sign(error(ind)) ~= sign(anterior)

if cont==0
error1=error(ind:length(error));
amplitud_min=min(error1)
amplitud=(amplitud_max-amplitud_min)/2
cont=1;
end

if error(ind) > 0
pos=[pos ind];
else
pos=[pos ind-1];
end
end
anterior=error(ind);
end

subplot(5,1,2)
plot(t,error,'.',t(pos),error(pos),'or'),grid on;
```

```
ylabel
('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
xlabel
('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
grid on;

plot(g_iteracion,t,error,'.',t(pos),error(pos),'or'),grid on;
title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,'Fontna
me','Courier New');
xlabel
(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,'Fontname'
,'Courier New');
grid (g_iteracion);

pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error

while(abs(wu-wc)>0.15)
w_total=[w_total wu]
L_total=[L_total L_rele]

L_rele=(wc-w_total(length(w_total))))/((w_total(length(w_total)))
(w_total(length(w_total)-1)))*((L_total(length(L_total)))-
(L_total(length(L_total)-1)))+(L_total(length(L_total)))
assignin('base', 'L_rele', L_rele);
sim('rele_delay_posicion')

relesinref=[rele(:,1),rele(:,2)];
t=relesinref(60:length(relesinref),1);

error=relesinref(60:length(relesinref),2);
amplitud_max=max(error)

pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
if sign(error(ind)) ~= sign(anterior)
if cont==0
error1=error(ind:length(error));
amplitud_min=min(error1)
```

```
amplitud=(amplitud_max-amplitud_min)/2
cont=1;
end
if error(ind) > 0
pos=[pos ind];
else
pos=[pos ind-1];
end
end
anterior=error(ind);
end
subplot(5,1,3)
plot(t,error,'.',t(pos),error(pos),'or'),grid on;
ylabel
('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
xlabel
('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
grid on;

plot(g_iteracion,t,error,'.',t(pos),error(pos),'or'),grid on;
title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,'Fontna
me','Courier New');
xlabel
(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,'Fontname'
,'Courier New');
grid (g_iteracion);

pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error
end

%%%llamada a la función magnitud_delay_int
[magnitud_estimada,magnitud_real,arg_estimado,arg_real]=Magnitud_D
elay_Int(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rele,grafica_
1,grafica_2,figura);

End
```

11.1.2. Test de relé velocidad

Se muestra toda la programación referente al test del relé con un sistema de velocidad.

```
function[amplitud,periodo_error,wu,L_rele,magnitud_estimada,magnit
ud_real,arg_estimado,arg_real]=Rele_Param_Delay(k,tau,L,wc,rango,d
,L_rele_1,L_rele_2,g_iteracion,grafica_1,grafica_2)

w_total=0;
L_total=0;

%L_rele=retardo primero
L_rele=L_rele_1;
assignin('base','L_rele',L_rele);%función que sirve para
introducir valores en workspace.
%wc=frecuencia de corte deseada
assignin('base', 'wc',wc);
%k=ganancia
assignin('base','k',k);
%tau=cte. de tiempo;
assignin('base','tau',tau);
%L=retardo planta;
assignin('base','L',L);
%rango=rango de frecuencias;
assignin('base', 'rango',rango);
%d=amplitud del rele
assignin('base', 'd',d);

%Barra de espera
bp=waitbar(0,'Program in execution. Please,wait...','Name','');

M=40;
I=0;

while I<M

I=I+1;
H=2*I;
G=I+3*I;
waitbar(I/M,bp);
sim('rele_delay')
end
delete(bp)

relesinref=[rele(:,1),rele(:,2)];

t=relesinref(60:length(relesinref),1);
error=relesinref(60:length(relesinref),2);

amplitud_max=max(error)
```

```
pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
if sign(error(ind)) ~= sign(anterior)

if cont==0
error1=error(ind:length(error));
amplitud_min=min(error1)
amplitud=(amplitud_max-amplitud_min)/2
cont=1;
end

if error(ind) > 0
pos=[pos ind];
else
pos=[pos ind-1];
end
end
anterior=error(ind);
end
figura=figure
subplot(5,1,1)
plot(t,error,'.',t(pos),error(pos),'or'),grid on;
title ('Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
xlabel
('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
grid on;
plot(g_iteracion,t,error,'.',t(pos),error(pos),'or'),grid on;
 title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,'Fontna
me','Courier New');
xlabel
(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,'Fontname'
,'Courier New');
grid (g_iteracion);

pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error
```

```
w_total=[w_total wu]
L_total=[L_total L_rele]

L_rele=L_rele_2;
assignin('base', 'L_rele', L_rele);

sim('rele_delay')

relesinref=[rele(:,1),rele(:,2)];
t=relesinref(60:length(relesinref),1);
error=relesinref(60:length(relesinref),2);

amplitud_max=max(error)

pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
if sign(error(ind)) ~= sign(anterior)

if cont==0
error1=error(ind:length(error));
amplitud_min=min(error1)
amplitud=(amplitud_max-amplitud_min)/2
cont=1;
end

if error(ind) > 0
pos=[pos ind];
else
pos=[pos ind-1];
end
end
anterior=error(ind);
end
subplot(5,1,2)
plot(t,error,'.',t(pos),error(pos),'or'),grid on;
ylabel
('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
xlabel
('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
grid on;
plot(g_iteracion,t,error,'.',t(pos),error(pos),'or'),grid on;
title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,'Fontna
me','Courier New');
xlabel
(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,'Fontname'
,'Courier New');
```

```
grid (g_iteracion);

pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error

while(abs(wu-wc)>0.15)
w_total=[w_total wu]
L_total=[L_total L_rele]


L_rele=(wc
(w_total(length(w_total))))/((w_total(length(w_total)))-
(w_total(length(w_total)-1)))*((L_total(length(L_total)))-
(L_total(length(L_total)-1)))+(L_total(length(L_total)))
assignin('base', 'L_rele', L_rele);
sim('rele_delay')

relesinref=[rele(:,1),rele(:,2)];

t=relesinref(60:length(relesinref),1);
error=relesinref(60:length(relesinref),2);

amplitud_max=max(error)
pos=[];
anterior=error(1);
cont=0;
for ind=2:length(error)
if sign(error(ind)) ~= sign(anterior)
if cont==0
error1=error(ind:length(error));
amplitud_min=min(error1)
amplitud=(amplitud_max-amplitud_min)/2
cont=1;
end
if error(ind) > 0
pos=[pos ind];
else
pos=[pos ind-1];
end
end
anterior=error(ind);
end
subplot(5,1,3)
plot(g_iteracion,t,error,'.',t(pos),error(pos),'or');
ylabel
('Amplitude','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
```

```
xlabel
('Time(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
grid on;
plot(g_iteracion,t,error,'.',t(pos),error(pos),'or');
title (g_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel
(g_iteracion,'Amplitude','Fontweight','Bold','Fontsize',11,'Fontna
me','Courier New');
xlabel
(g_iteracion,'Time(s)','Fontweight','Bold','Fontsize',9,'Fontname'
,'Courier New');
grid (g_iteracion);
pos_periodo=pos(1:2:length(pos));
periodo=0;
cont=0;
for r=2:length(pos_periodo)
periodo=periodo+(t(pos_periodo(r))-t(pos_periodo(r-1)));
cont=cont+1;
end
periodo_error=periodo/cont
frecuencia_error=1/periodo_error
wu=2*pi*frecuencia_error
end
%%%llamada a la funcion magnitud_delay
[magnitud_estimada,magnitud_real,arg_estimado,arg_real]=Magnitud_D
elay(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rele,grafica_1,gr
afica_2,figura);


End
```

11.1.3. Estimación frecuencial posición

Se explica la programación utilizada para la estimación frecuencial de un sistema de posición.

```
function[magnitud_estimada_db,magnitud_real_db,argumento_estimado,
argumento_real]=
Magnitud_Delay_Int(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rel
e,grafica_1,grafica_2,figura)

%función de tranferencia de la planta
ret=exp(-L*j*(2*pi*frecuencia_error));
num=k*ret;
den=polyval([tau 1 0],(j*(2*pi*frecuencia_error)));
```

```
total=num./den;
rango=2;
w=logspace(-rango+log10(wc),rango+log10(wc),200);
ret_real=exp(-L*j*w);
num_real=k*ret_real;
den_real=polyval([tau 1 0],(j*w));
total_real=num_real./den_real;
magnitud_real=abs(total);
magnitud_real_db=20*log10(magnitud_real)

magnitud_estimada=pi*amplitud/(4*d);
magnitud_estimada_db=20*log10(magnitud_estimada)
%argumento_real=(-atan(tau*2*pi*frecuencia_error))*180/pi
argumento_real=angle(total)*180/pi
argumento_estimado=((wu*L_rele)-pi)*180/pi
%atraso_adelanto;
%calculo_PI;

mgtotal_real=20*log10(abs(total_real));ftotal_real=(180/pi)*angle(
total_real);
subplot(5,1,4)
r=2;
semilogx(w,mgtotal_real);
title ('Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel('Magnitude
(dB)','Fontweight','Bold','Fontsize',11,'Fontname','Courier New');
grid on;

semilogx(grafica_1,w,mgtotal_real);
title (grafica_1,'Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel(grafica_1,'Magnitude(dB)','Fontweight','Bold','Fontsize',11
,'Fontname','Courier New');
grid (grafica_1);

subplot(5,1,5)
semilogx(w,ftotal_real)
grid;
xlabel('Frequency(rad/s)','Fontweight','Bold','Fontsize',9,'Fontna
me','Courier New');
ylabel('Phase(deg)','Fontweight','Bold','Fontsize',11,'Fontname','
Courier New');
grid on;
saveas(figura,'graficas.fig')
set(figura,'visible','Off')
semilogx(grafica_2,w,ftotal_real)
grid;
xlabel(grafica_2,'Frequency
(rad/s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
```

```
ylabel(grafica_2,'Phase
(deg)','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
grid (grafica_2);

end
```

11.1.4. Estimación frecuencial velocidad

Se detalla la programación utilizada para la estimación frecuencial de un sistema de velocidad.

```
function[magnitud_estimada_db,magnitud_real_db,argumento_estimado,
argumento_real]=
Magnitud_Delay(L,k,tau,frecuencia_error,wu,wc,amplitud,d,L_rele,gr
afica_1,grafica_2,figura)

%función de tranferencia de la planta
ret=exp(-L*j*(2*pi*frecuencia_error));
num=k*ret;
den=polyval([tau 1 0],(j*(2*pi*frecuencia_error)));

total=num./den;
rango=2;
w=logspace(-rango+log10(wc),rango+log10(wc),200);
ret_real=exp(-L*j*w);
num_real=k*ret_real;
den_real=polyval([tau 1 0],(j*w));
total_real=num_real./den_real;
magnitud_real=abs(total);
magnitud_real_db=20*log10(magnitud_real)

magnitud_estimada=pi*amplitud/(4*d);
magnitud_estimada_db=20*log10(magnitud_estimada)
%argumento_real=(-atan(tau*2*pi*frecuencia_error))*180/pi
argumento_real=angle(total)*180/pi
argumento_estimado=((wu*L_rele)-pi)*180/pi
%atraso_adelanto;
%calculo_PI;

mgtotal_real=20*log10(abs(total_real));ftotal_real=(180/pi)*angle(
total_real);
subplot(5,1,4)
r=2;
semilogx(w,mgtotal_real);
title ('Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel('Magnitude
(dB)','Fontweight','Bold','Fontsize',11,'Fontname','Courier New');
grid on;
```

```
semilogx(grafica_1,w,mgtotal_real);
title (grafica_1,'Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
ylabel(grafica_1,'Magnitude(dB)','Fontweight','Bold','Fontsize',11
,'Fontname','Courier New');
grid (grafica_1);

subplot(5,1,5)
semilogx(w,ftotal_real)
grid;
xlabel('Frequency(rad/s)','Fontweight','Bold','Fontsize',9,'Fontna
me','Courier New');
ylabel('Phase(deg)','Fontweight','Bold','Fontsize',11,'Fontname','
Courier New');
grid on;
saveas(figura,'graficas.fig')
set(figura,'visible','Off')

semilogx(grafica_2,w,ftotal_real)
grid;
xlabel(grafica_2,'Frequency
(rad/s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier
New');
ylabel(grafica_2,'Phase
(deg)','Fontweight','Bold','Fontsize',11,'Fontname','Courier
New');
grid (grafica_2);

end
```

11.1.5. Programación interfaz

En este apartado se detalla toda la programación necesaria para la creación de la interfaz diseñada.

```
function main
%%%%%%%%%%%%%----variables globales----%%%%%%%%%%%%

global ANCHO
global ALTO
global espera
global interfaz
global EJE
global S
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

screensize = get(0,'ScreenSize');% get te saca los valores de
fabrica de la pantalla de matlaba[left,bottom.width,height]
ANCHO = screensize (3);
ALTO = screensize (4);

%%%%%%%%%%%%----Figura de espera ----%%%%%%%%%%%%%
espera = figure(...
'Name','',...
'NumberTitle','off', ...
'Menubar','none',...
'Units','pixels',...
'Position',[0,0,ANCHO,ALTO],...
'Color',[0.078,0.169,0.549]);

uicontrol(...
'Parent',espera,...
'Units','normalized','Position',[0.06,0.45,0.8,0.2],...
'Style','Text','BackgroundColor',[0.078,0.169,0.549],...
'Fontname','Courier New','Fontsize',32,'Fontweight','Bold',...
'ForeGroundColor',[1 1 1],'FontAngle','normal',...
'String','Loading interface...');

maximize(espera)%función que adapta el programa a la pantalla

%%%%%%%%%%%%----Figura principal del programa----%%%%%%%%%%%%%

interfaz = figure(...
'Visible','off',...
'Name','',...
'NumberTitle','off', ..
'Menubar','none',...
'Units','normalized',...
'Position',[0,0,1,1]);

uitabpanel(...
'Parent',interfaz,...
'TabPosition','lefttop',...
'Units','Pixels',...
'Position',[0,0,1.1*ANCHO,0.93*ALTO],...
'PanelBorderType','line',...
'Title',{'   IDENTIFICATION   ',},...
'BackgroundColor',[0.078,0.169,0.549],...
'CreateFcn',@CreateTab);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%-------------------------- FUNCTION CREATE TAB 1 ---------------
--------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
```

```
function CreateTab(varargin)

[htab,hpanel,hstatus] = varargin{[1,3,4]};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %------------------------Panel entrada-------------------------%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

E.panel_entrada=uipanel(...
'Parent',hpanel(1),'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.02 0.44 0.4 0.54],...
'BackgroundColor',[0.82,0.863,0.922]);

%Encabezado panel entrada
E.texto_encabezado_entrada= uicontrol(...
'Parent',E.panel_entrada,'Units','normalized',...
'Position',[0.01,0.94,0.98,0.05],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','INPUT DATA',...
'Fontname','Courier New','Fontsize',14,...
'Fontweight','Bold',...
'HorizontalAlignment','Center');

%panel para elegir el sistema
E.panel_dibujos=uipanel(...
'Parent',E.panel_entrada,'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.03,0.65,0.94,0.28],...
'BackgroundColor',[1 1 1]);
axes(...
'Parent',E.panel_dibujos,...
'Units','pixel',...
'Position',[10,6,522,65]);

b=imread('dibujos.jpg');
imshow(b),axis off,hold on

%checkbox posición
E.concatenar_1 = uicontrol(...
'Parent',E.panel_dibujos,'Units','Normalized',...
'Style','checkbox','Enable','On',...
'BackgroundColor',[1,1,1],...
'Position',[0.05,0.76,0.3,0.17],...
'Fontname','Courier New','Fontsize',10,'String','Position
System',...
'Callback',@Check_Callback);

%checkbox velocidad
E.concatenar_2 = uicontrol(...
'Parent',E.panel_dibujos,'Units','Normalized',...
'Style','checkbox','Enable','On',...
'BackgroundColor',[1,1,1],...
'Position',[0.62,0.76,0.35,0.17],...
'Fontname','Courier New','Fontsize',10,'String','Velocity
System',...
```

```
'Callback',@Check_Callback);

%%%%%%-----subfunción para elegir el tipo de sistema-%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Check_Callback(hObject, eventdata, handles)

set( E.boton_new,'Enable','Off');

if hObject == E.concatenar_
if get(E.concatenar_1,'Value') == 1

set(E.concatenar_2,'value',0);
set(E.dat_ganancia,'Enable','On');
set(E.dat_cte.tiempo,'Enable','On');
set(E.retardo,'Enable','On');
set(E.amplitud_rele,'Enable','On');
set(E.dat_frecuencia,'Enable','On');
set(E.dat_rango,'Enable','On');
set(E.retardo_1,'Enable','On');
set(E.retardo_2,'Enable','On');
set( E.boton_new,'Enable','Off');

%botón para activar el programa.
E.boton_run= uicontrol (...
'Parent',E.panel_entrada,'Units','Normalized',...
'Position',[0.29,0.01,0.65,0.08],...
'Style','pushbutton','Enable','On',...
'String','Run',...
'Fontname','Courier New','Fontsize',13,...
'Callback',@Position_Data_Callback);

elseif get(E.concatenar_1,'Value') == 0

set(E.dat_ganancia,'Enable','Off');
set(E.dat_cte.tiempo,'Enable','Off');
set(E.retardo,'Enable','Off');
set(E.amplitud_rele,'Enable','Off');
set(E.dat_frecuencia,'Enable','Off');
set(E.dat_rango,'Enable','Off');
set(E.retardo_1,'Enable','Off');
set(E.retardo_2,'Enable','Off');
set(E.boton_run,'Enable','Off');
end

elseif hObject == E.concatenar_2
if get(E.concatenar_2,'Value') == 1

set(E.concatenar_1,'value',0);
set(E.dat_ganancia,'Enable','On');
set(E.dat_cte.tiempo,'Enable','On');
set(E.retardo,'Enable','On');
set(E.amplitud_rele,'Enable','On');
set(E.dat_frecuencia,'Enable','On');
```

```
set(E.dat_rango,'Enable','On');
set(E.retardo_1,'Enable','On');
set(E.retardo_2,'Enable','On');

E.boton_run= uicontrol (...
'Parent',E.panel_entrada,'Units','Normalized',...
'Position',[0.29,0.01,0.65,0.08],...
'Style','pushbutton','Enable','On',...
'String','Run',...
'Fontname','Courier New','Fontsize',13,...
'Callback',@Velocity_Data_Callback);
elseif get(E.concatenar_2,'Value') == 0
set(E.dat_ganancia,'Enable','Off');
set(E.dat_cte.tiempo,'Enable','Off');
set(E.retardo,'Enable','Off');
set(E.amplitud_rele,'Enable','Off');
set(E.dat_frecuencia,'Enable','Off');
set(E.dat_rango,'Enable','Off');
set(E.retardo_1,'Enable','Off');
set(E.retardo_2,'Enable','Off');
set(E.boton_run,'Enable','Off');
end

end

end

E.subpanel_entrada =uipanel(...
'Parent',E.panel_entrada,'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.03,0.1,0.360,0.53],...
'BackgroundColor',[0.82,0.863,0.922]);
E.texto_entrada= uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.01,0.89,0.98,0.07],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','PLANT DATA',...
'Fontname','Courier New','Fontsize',10,...
'TooltipString',...
'Información sobre el botón',...
'Fontweight','Bold',...
'HorizontalAlignment','Center');


%Cuadros de texto

E.control_text(1) = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.02,0.55,0.70,0.2],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.control_text(2) = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.02,0.3,0.76,0.20],...
'BackgroundColor',[0.82,0.863,0.922],...
```

```
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.control_text(3) = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.02,0.05,0.70,0.2],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

set(E.control_text(1),'String','Gain:');
set(E.control_text(2),'String','Time Constant (s):');
set(E.control_text(3),'String','Delay (s):');

%Cajas para introducir datos de la planta
E.dat_ganancia = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.75,0.60,0.23,0.20],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');

E.dat_cte.tiempo = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.75,0.35,0.23,0.20],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');
E.retardo = uicontrol(...
'Parent',E.subpanel_entrada,'Units','normalized',...
'Position',[0.75,0.10,0.23,0.20],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');

E.subpanel_test =uipanel(...
'Parent',E.panel_entrada,'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.41,0.1,0.56,0.53],...
'BackgroundColor',[0.82,0.863,0.922]);
E.texto_entrada= uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.01,0.89,0.98,0.07],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','RELAY TEST DATA',...
'Fontname','Courier New','Fontsize',10,...
'Fontweight','Bold',...
'HorizontalAlignment','Center');

%Cuadros de texto del test

E.control_text(4) = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.02,0.69,0.70,0.14],...
'BackgroundColor',[0.82,0.863,0.922],...
```

```
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.control_text(5) = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.02,0.525,0.80,0.14],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.control_text(6) = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.02,0.36,0.70,0.14],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.control_text(7) = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.02,0.195,0.70,0.14],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.control_text(8) = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.02,0.03,0.70,0.14],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','text','Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

set(E.control_text(4),'String','User frequency  (rad/s):');
set(E.control_text(5),'String','Frequency range (decades):');
set(E.control_text(6),'String','Relay amplitude:');
set(E.control_text(7),'String','First delay (s):');
set(E.control_text(8),'String','Second delay (s):');


%Cajas para introducir datos del test

E.dat_frecuencia = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.83,0.71,0.14,0.14],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');
E.dat_rango = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.83,0.545,0.14,0.14],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');
E.amplitud_rele = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.83,0.38,0.14,0.14],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
```

```
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');
E.retardo_1 = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.83,0.215,0.14,0.14],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');
E.retardo_2 = uicontrol(...
'Parent',E.subpanel_test,'Units','normalized',...
'Position',[0.83,0.05,0.14,0.14],...
'Style','edit','Fontname','CourierNew','Fontsize',10,'Enable','Off
',...
'BackgroundColor',[1,1,1],...
'HorizontalAlignment','Center');

 E.boton_run= uicontrol (...
'Parent',E.panel_entrada,'Units','Normalized',...
'Position',[0.29,0.01,0.65,0.08],...
'Style','pushbutton','Enable','Off',...
'String','Run',...
'Fontname','Courier New','Fontsize',13);
E.boton_new= uicontrol (...
'Parent',E.panel_entrada,'Units','Normalized',...
'Position',[0.05,0.01,0.2,0.08],...
'Style','pushbutton','Enable','Off',...
'String','New Data',...
'Fontname','Courier New','Fontsize',12);

%subfunción para cargar los datos al programa

function Position_Data_Callback(hObject, eventdata, handles)

E.string_ganancia=get(E.dat_ganancia,'string');
E.string_cte.tiempo=get( E.dat_cte.tiempo,'string');
E.string_retardo=get(E.retardo,'string');
E.string_frecuencia=get(E.dat_frecuencia,'string');
E.string_rango=get(E.dat_rango,'string');
E.string_amplitud_rele=get( E.amplitud_rele,'string');
E.string_retardo_1=get( E.retardo_1 ,'string');
E.string_retardo_2=get( E.retardo_2 ,'string');

%compara que no haya ninguna caja vacía
if strcmp('',E.string_ganancia) ||
strcmp('',E.string_cte.tiempo)|| strcmp('',E.string_retardo)||...
strcmp('',E.string_rango)||strcmp('', E.string_frecuencia)||
strcmp('',
E.string_amplitud_rele)||strcmp('',E.string_retardo_1)||
strcmp('',E.string_retardo_2)
msgbox('Input data missing.Please check','Error','error');

else

S.Gain = str2num(E.string_ganancia);
```

```
S.Time_constant = str2num(E.string_cte.tiempo);
S.Delay = str2num(E.string_retardo);
S.User_frequency = str2num(E.string_frecuencia);
S.Frequency_range = str2num(E.string_rango);
S.Relay_amplitude =str2num(E.string_amplitud_rele);
S.First_delay = str2num(E.string_retardo_1);
S.Second_delay = str2num(E.string_retardo_2);


%%%%% ¡¡¡¡¡Llamada a la función rele_param_delay_pos !!!!!%%%%%%%
[S.Output_amplitude,S.Output_period,S.Output_frequency,S.Final_del
ay,S.Magnitude_estimated,...

S.Magnitude_real,S.Phase_estimated,S.Phase_real]=Rele_Param_Delay_
Pos(S.Gain,S.Time_constant,S.Delay,...

S.User_frequency,S.Frequency_range,S.Relay_amplitude,S.First_delay
,S.Second_delay,...

EJE.grafica_iteracion,EJE.grafica_bode_magnitud,EJE.grafica_bode_f
ase);

%conversion de variables que vienen de la función de num a string

E.conv_amplitud= num2str(S.Output_amplitude,4);
E.conv_periodo= num2str(S.Output_period,4);
E.conv_frecuencia_1= num2str(S.Output_frequency,4);
E.conv_magnitud= num2str(S.Magnitude_estimated,4);
E.conv_magnitud_1= num2str(S.Magnitude_real,4);
E.conv_fase= num2str(S.Phase_estimated,4);
E.conv_fase_1= num2str(S.Phase_real,4);

%llamada a una subfunción
Display_Data ;
end
end

function Velocity_Data_Callback(hObject, eventdata, handles)

E.Gain_Vel=get(E.dat_ganancia,'string');
E.Time_constant_Vel=get( E.dat_cte.tiempo,'string');
E.Delay_Vel=get(E.retardo,'string');
E.Frequency_Vel=get(E.dat_frecuencia,'string');
E.Frequency_Range_Vel=get(E.dat_rango,'string');
E.Relay_Amplitude_Vel=get( E.amplitud_rele,'string');
E.Delay_1_Vel=get( E.retardo_1 ,'string');
E.Delay_2_Vel=get( E.retardo_2 ,'string');

%compara que no haya ninguna caja vacía
if strcmp('',E.Gain_Vel) || strcmp('',E.Time_constant_Vel)||
strcmp('',E.Delay_Vel)||...
strcmp('',E.Frequency_Range_Vel)||strcmp('', E.Frequency_Vel)||
strcmp('', E.Relay_Amplitude_Vel)||strcmp('',E.Delay_1_Vel)||
strcmp('',E.Delay_2_Vel)
msgbox('Input data missing.Please check','Error','error');
```

```
else
clear S;
S.Gain_Vel = str2num(E.Gain_Vel);
S.Time_constant_Vel = str2num(E.Time_constant_Vel);
S.Delay_Vel = str2num(E.Delay_Vel);
S.Frequency_Vel = str2num(E.Frequency_Vel);
S.Frequency_Range_Vel = str2num(E.Frequency_Range_Vel);
S.Relay_Amplitude_Vel=str2num(E.Relay_Amplitude_Vel);
S.Delay_1_Vel = str2num(E.Delay_1_Vel);
S.Delay_2_Vel = str2num(E.Delay_2_Vel);

%%%%% ¡¡¡¡¡Llamada a la función rele_param_delay!!!!%%%%%%%
[S.Output_amplitude,S.Output_period,S.Output_frequency,S.Final_del
ay,S.Magnitude_estimated,...

S.Magnitude_real,S.Phase_estimated,S.Phase_real]=Rele_Param_Delay(
S.Gain_Vel,S.Time_constant_Vel,S.Delay_Vel,...
S.Frequency_Vel,S.Frequency_Range_Vel,S.Relay_Amplitude_Vel,S.Dela
y_1_Vel ,S.Delay_2_Vel,...

EJE.grafica_iteracion,EJE.grafica_bode_magnitud,EJE.grafica_bode_f
ase);

%conversion de variables que vienen de la funcion de num a string
E.conv_amplitud= num2str(S.Output_amplitude);
E.conv_periodo= num2str(S.Output_period);
E.conv_frecuencia_1= num2str(S.Output_frequency);
E.conv_delay= num2str(S.Final_delay);
E.conv_magnitud= num2str(S.Magnitude_estimated);
E.conv_magnitud_1= num2str(S.Magnitude_real);
E.conv_fase= num2str(S.Phase_estimated);
E.conv_fase_1= num2str(S.Phase_real);
Display_Data;
end
end

E.panel_salida=uipanel(...
'Parent',hpanel(1),'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.02 0.02 0.25 0.4],...
'BackgroundColor',[0.82,0.863,0.922]);

% Cuadro texto rele test results
E.texto_salida=uicontrol(...
'Parent',E.panel_salida,'Units','normalized',...
'Position',[0.01,0.92,0.98,0.07],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','RELAY TEST RESULTS',...
'Fontname','Courier New','Fontsize',13,...
'Fontweight','Bold',...
'HorizontalAlignment','Center');

E.subpanel_salida =uipanel(...
'Parent',E.panel_salida,'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.04,0.40,0.92,0.5],...
```

```
'BackgroundColor',[0.82,0.863,0.922]);

 %------  Amplitude --------%
E.texto_amplitud_salida=uicontrol(...
'Parent',E.subpanel_salida,'Units','normalized',...
'Position',[0.02,0.73,0.6,0.17],...
'Style','Text','String','Output amplitude:',...
'Fontname','Courier New','Fontsize',10,...
'BackgroundColor',[0.82,0.863,0.922],...
'HorizontalAlignment','Left');

E.sal_amplitud=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.76,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');

%------  Period --------%
E.texto_periodo_salida=uicontrol(...
'Parent',E.subpanel_salida,'Units','normalized',...
'Position',[0.02,0.5,0.6,0.17],...
'Style','Text','String','Output period (s):',...
'Fontname','Courier New','Fontsize',10,...
'BackgroundColor',[0.82,0.863,0.922],...
'HorizontalAlignment','Left');

E.sal_periodo=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.53,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');
%------  Frequency --------%
E.texto_frecuencia=uicontrol(...
'Parent',E.subpanel_salida,'Units','normalized',...
'Position',[0.02,0.27,0.66,0.17],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','Output frequency (rad/s):',...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

E.sal_frecuencia=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.30,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');
%------  Delay(s)--------%
```

```
E.texto_delay=uicontrol(...
'Parent',E.subpanel_salida,'Units','normalized',...
'Position',[0.02,0.04,0.63,0.17],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','Final delay (s):',...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

E.sal_delay=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.07,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');

E.subpanel_bode =uipanel(...
'Parent',E.panel_salida,'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.04,0.02,0.92,0.36],...
'BackgroundColor',[0.82,0.863,0.922]);
    %
E.texto_estimados=uicontrol(...
'Parent',E.subpanel_bode,'Units','normalized',...
'Position',[0.52,0.8,0.3,0.14],...
'Style','Text','String','Estimated:',...
'BackgroundColor',[0.82,0.863,0.922],...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');
E.texto_estimados=uicontrol(...
'Parent',E.subpanel_bode,'Units','normalized',...
'Position',[0.81,0.8,0.15,0.14],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','Real:',...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

%----------Magnitude-------------%

E.texto_magnitud=uicontrol(...
'Parent',E.subpanel_bode,'Units','normalized',...
'Position',[0.02,0.55,0.51,0.2],...
'Style','Text','String','Magnitude (dB):',...
'BackgroundColor',[0.82,0.863,0.922],...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

E.sal_magnitud=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.52,0.49,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');
```

```
E.sal_magnitud_1=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.8,0.47,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');

 %-----------Phase-------------%
E.texto_fase=uicontrol(...
'Parent',E.subpanel_bode,'Units','normalized',...
'Position',[0.02,0.15,0.51,0.2],...
'BackgroundColor',[0.82,0.863,0.922],...
'Style','Text','String','Phase (deg):',...
'Fontname','Courier New','Fontsize',10,...
'HorizontalAlignment','Left');

E.sal_fase=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.52,0.1,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');
E.sal_fase_1=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.8,0.1,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center');

% Panel fondo graficas
E.panel_salida=uipanel(...
'Parent',hpanel(1),'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.44 0.01 0.46 0.98],...
'BackgroundColor',[0.82,0.863,0.922]);

%%% gráfica iteración %%%
EJE.grafica_iteracion=axes(...
'Parent',hpanel(1),...
'Position',[0.485 0.71 0.4 0.23]);

title (EJE.grafica_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
axis([0 60 -10 10])
ylabel (EJE.grafica_iteracion,'Amplitude
','Fontweight','Bold','Fontsize',11,'Fontname','Courier New');
xlabel (EJE.grafica_iteracion,'Time
(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier New');
grid on;
```

```
%%% gráfica magnitud %%%
EJE.grafica_bode_magnitud=axes(...
'Parent',hpanel(1),...
'Position',[0.485,0.38,0.4,0.23]);
x = 1000;
y = -50:10:100;
semilogx(x,y);
ylabel ( EJE.grafica_bode_magnitud,'Magnitude
(dB)','Fontweight','Bold','Fontsize',11,'Fontname','Courier New');
grid on;
title (EJE.grafica_bode_magnitud,'Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');

%%% gráfica fase %%%
EJE.grafica_bode_fase=axes(...
'Parent',hpanel(1),...
'Position',[0.485,0.085,0.4,0.23]);
x = 1000;
y = -50:10:100;
semilogx(x,y);
xlabel (
EJE.grafica_bode_fase,'Frequency(rad/s)','Fontweight','Bold','Font
size',9,'Fontname','Courier New');
ylabel (
EJE.grafica_bode_fase,'Phase(deg)','Fontweight','Bold','Fontsize',
11,'Fontname','Courier New');
grid on;

%%%%botón para guardar las variables en un archivo.mat %%%%
E.boton_save= uicontrol (...
'Parent',hpanel(1),'Units','Normalized',...
'Position',[0.28,0.05,0.15,0.05],...
'Style','pushbutton','Enable','On',...
'String','Save experiment data(.mat)',...
'Fontname','Courier New','Fontsize',10,'callback',@Save_Data);

%%%%botón para guardar una imagen de la pantalla %%%%
 E.boton_save_screen= uicontrol (...
'Parent',hpanel(1),'Units','Normalized',...
'Position',[0.28,0.12,0.15,0.05],...
'Style','pushbutton','Enable','On',...
'String','Save screenshot',...
'Fontname','CourierNew','Fontsize',10,'callback',@Save_Interface);

%%%%botón de ayuda%%%%

E.boton_ayuda= uicontrol (...
'Parent',hpanel(1),'Units','Normalized',...
'Position',[0.31,0.22,0.07,0.07],...
'Style','pushbutton','Enable','On',...
'String','HELP',...
'Fontname','Courier New','Fontsize',11,'callback',@Read_PDF);
```

```
%%%%%%%%%EJES para logotipo%%%%%%%%%
axes(...
'Parent',hpanel(1),...
'Units','Normalized',...
'Position',[0.28,0.32,0.14,0.1]);

a=imread('Logo.png');
imshow(a),axis off,hold on


function Display_Data ()
%paso de las variables a los edit
E.sal_amplitud=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.76,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string', E.conv_amplitud);
E.sal_periodo=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.53,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...

'HorizontalAlignment','Center','string',E.conv_periodo);
E.sal_frecuencia=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.30,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string',  E.conv_frecuencia_1);
E.sal_delay=uicontrol(...
'Parent',E.subpanel_salida,...
'Units','normalized',...
'Position',[0.72,0.07,0.21,0.20],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string',E.conv_delay);
E.sal_magnitud=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.52,0.49,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string',   E.conv_magnitud);
E.sal_magnitud_1=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
```

```
'Position',[0.8,0.47,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string', E.conv_magnitud_1);
E.sal_fase=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.52,0.1,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string', E.conv_fase);
E.sal_fase_1=uicontrol(...
'Parent',E.subpanel_bode,...
'Units','normalized',...
'Position',[0.8,0.1,0.19,0.3],...
'Style','text','Fontname','Courier New','Fontsize',10,...

'BackgroundColor',[0.859,0.843,0.808],'Enable','On',...
'HorizontalAlignment','Center','string', E.conv_fase_1);

E.boton_new= uicontrol (...
'Parent',E.panel_entrada,'Units','Normalized',...
'Position',[0.05,0.01,0.2,0.08],...
'Style','pushbutton','Enable','On',...
'String','New Data',...
'Fontname','Courier New','Fontsize',12,...
'callback',@New_data);
set(E.boton_run,'enable','off');
set(E.concatenar_1,'enable','off');
set(E.concatenar_2,'enable','off');

end

function New_data(hObject, eventdata, handles)

set(E.concatenar_1,'enable','on');
set(E.concatenar_2,'enable','on');
set(E.dat_ganancia,'String','','enable','off');
set(E.dat_cte.tiempo,'String','','enable','off');
set(E.retardo,'String','','enable','off');
set(E.amplitud_rele,'String','','enable','off');
set(E.dat_frecuencia,'String','','enable','off');
set(E.dat_rango,'String','','enable','off');
set(E.retardo_1,'String','','enable','off');
set(E.retardo_2,'String','','enable','off');
set( E.sal_amplitud,'String','','enable','off');
set(E.sal_periodo,'String','','enable','off');
set(E.sal_frecuencia,'String','','enable','off');
set(  E.sal_delay,'String','','enable','off');
set(E.sal_magnitud,'String','','enable','off');
set(  E.sal_magnitud_1,'String','','enable','off');
set(E.sal_fase,'String','','enable','off');
set(E.sal_fase_1,'String','','enable','off');
```

```
set(E.concatenar_1,'value',0);
set(E.concatenar_2,'value',0);
set(E.boton_run,'enable','off');
% Panel fondo graficas
E.panel_salida=uipanel(...
'Parent',hpanel(1),'ForeGroundColor',[1,1,1],...
'Units','normalized','Position',[0.44 0.01 0.46 0.98],...
'BackgroundColor',[0.82,0.863,0.922]);

%%% gráfica iteración %%%
EJE.grafica_iteracion=axes(...
'Parent',hpanel(1),...
'Position',[0.485 0.71 0.4 0.23]);

title (EJE.grafica_iteracion,'Temporal Response of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');
axis([0 60 -10 10])
ylabel
(EJE.grafica_iteracion,'Amplitude','Fontweight','Bold','Fontsize',
11,'Fontname','Courier New');
xlabel (EJE.grafica_iteracion,'Time
(s)','Fontweight','Bold','Fontsize',9,'Fontname','Courier New');
grid on;

%%% gráfica magnitud %%%
EJE.grafica_bode_magnitud=axes(...
'Parent',hpanel(1),...
'Position',[0.485,0.38,0.4,0.23]);
x = 1000;
y = -50:10:100;
semilogx(x,y);
ylabel ( EJE.grafica_bode_magnitud,'Magnitude
(dB)','Fontweight','Bold','Fontsize',11,'Fontname','Courier New');
grid on;
title (EJE.grafica_bode_magnitud,'Bode Diagram of the
System','Fontweight','Bold','Fontsize',14,'Fontname','Courier
New');

%%% gráfica fase %%%
EJE.grafica_bode_fase=axes(...
'Parent',hpanel(1),...
'Position',[0.485,0.085,0.4,0.23]);
x = 1000;
y = -50:10:100;
semilogx(x,y);
xlabel (
EJE.grafica_bode_fase,'Frequency(rad/s)','Fontweight','Bold','Font
size',9,'Fontname','Courier New');
ylabel (
EJE.grafica_bode_fase,'Phase(deg)','Fontweight','Bold','Fontsize',
11,'Fontname','Courier New');
grid on;
end
```

```
%subfunción para guardar las variables en un archivo.mat

function Save_Data(hObject, eventdata, handles)

[filename, pathname] = uiputfile('*.mat');
save(filename,'*S')
clear *S;
end

%subfunción para guardar una imagen de la pantalla

function Save_Interface(varargin)
% get(0,'CurrentFigure');
[filename, pathname] = uiputfile('*.jpg');
hgexport(gcf, filename,hgexport('factorystyle'), 'Format',
'jpeg');
end
%subfuncion para mostrar la ayuda
function Read_PDF (varargin)
open('Help.pdf');
end
end
set(interfaz,'visible','On')
maximize(interfaz)
delete(espera)
end
```

11.2. ANEXO 2: DOCUMENTO DE AYUDA

En este anexo se adjunta el documento de ayuda que detalla todos los pasos que el usuario debe hacer para el correcto uso de la interfaz. Se trata de un documento de tipo PDF que se abre a partir del botón de "Help" incluido en la interfaz.

SYSTEMS IDENTIFICATION BY USING THE MODIFIED RELAY TEST

Here some guidelines are summarized for the use of the identification toolbox:

1. In the "Input data" panel, choose the type of dynamic system to be identified. The modified relay test method has two different options: Position system or Velocity system, user has to choose one of those, to start identifying the system.

2. Fill input data out, which are enabled after system is chosen on first step. Do not forget to take into account units of each variable: s→seconds; rad→radians; decades→ decades.

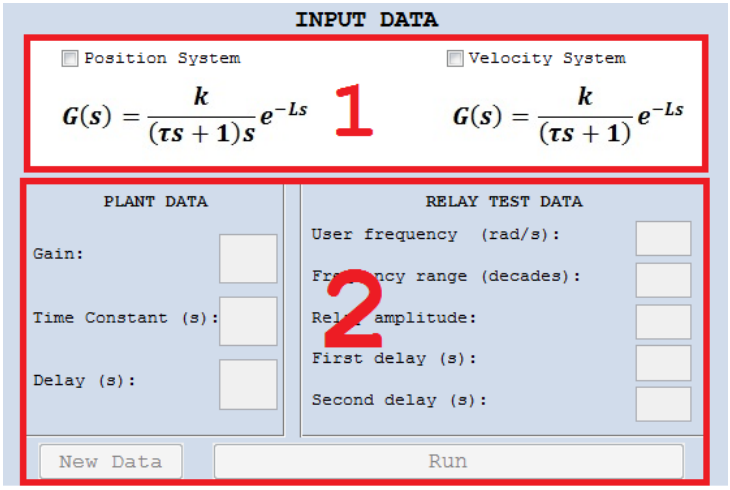


Figure 24 : "Input Data" Panel.

3. Click on **Run** button to continue. While any of input data is incomplete, next error message will appear: "Input data missing. Please check".

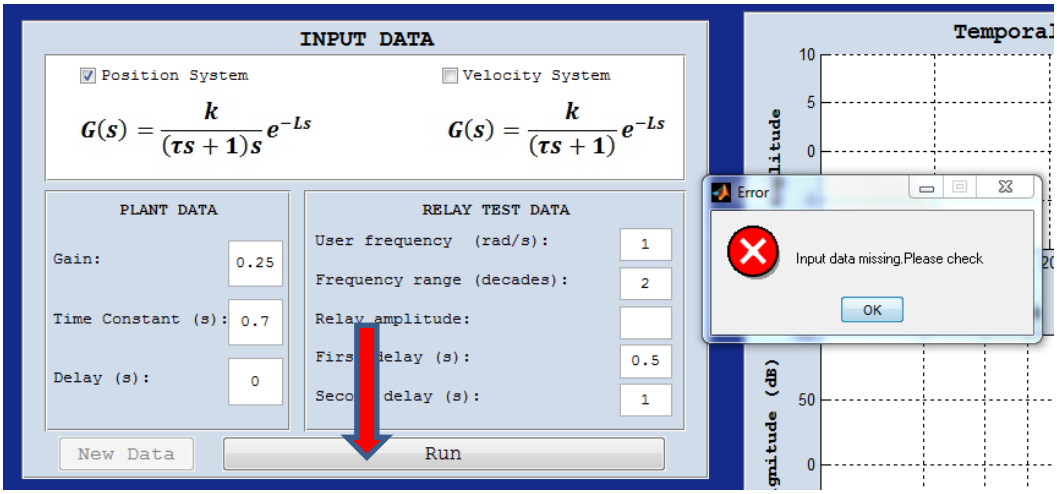


Figure 25: Error message.

4. Wait until process is successful, timeout is indicated on next window.

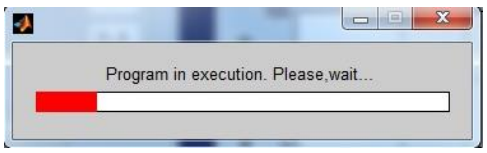


Figure 26: Waiting window.

5. Analyze obtained results after process is done. Results are shown on “Relay test results” panel. It can be seen that there are two different magnitude and phase variables: **estimated value,** which is the result of using modified relay method, and **real value,** which is obtained theoretically. Remember to take into account units of each variable: Db→decibels; deg→degrees.

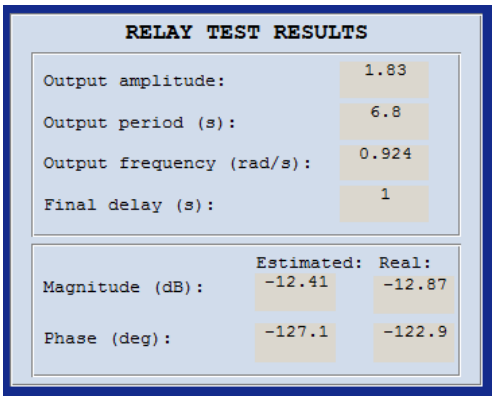


Figure 27: "Relay Test Results" panel.

6. Observe the different graphs. The first one corresponds to temporal response of last itineration process. Second and third ones are Bode’s plots obtained with the results.

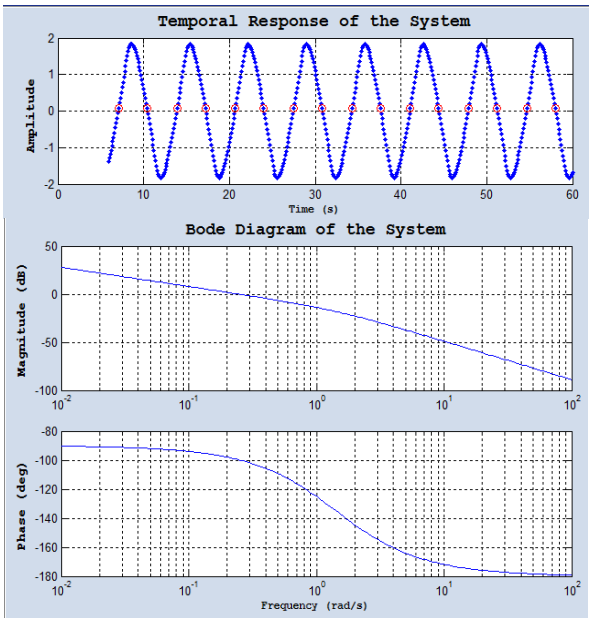


Figure 28: Graphs.

7. Save the results with the different buttons:

a. Save screenshot button: it saves image of whole screen.
b. Save experiment data (.mat) button: it saves inputs and outputs data in a .mat type file.

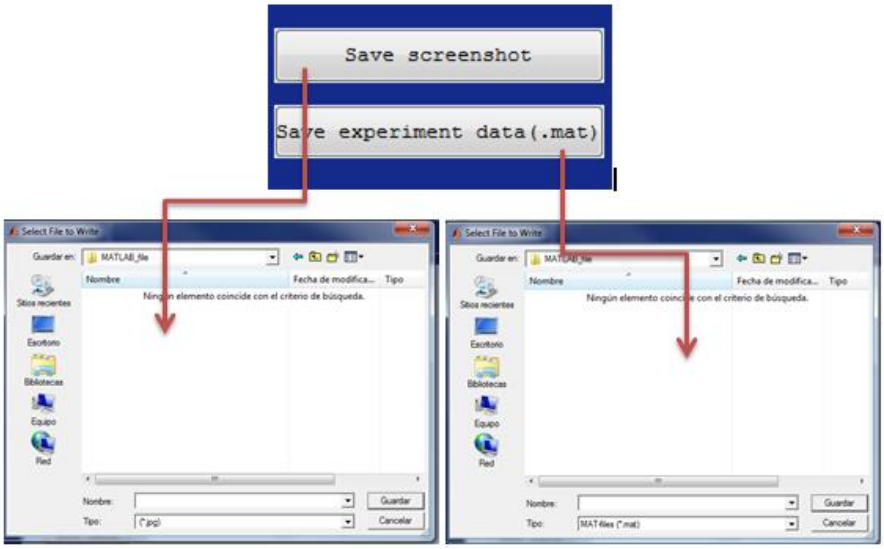


Figure 29: Save Buttons.

Remember that every graph is saved automatically in a .fig file.

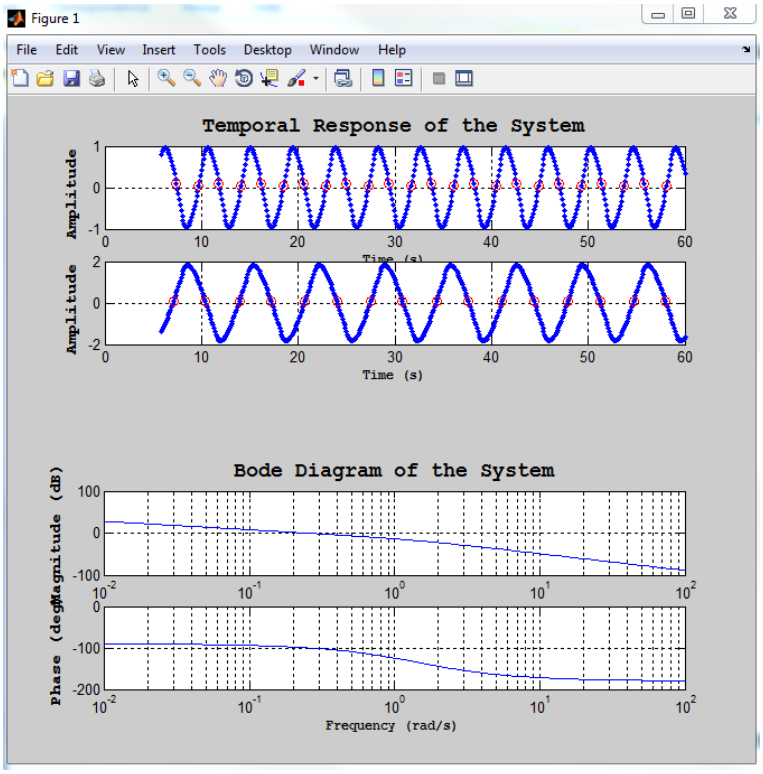


Figura 30: Graphs.fig file.

8. Start a new process with **new data** button or close the program.

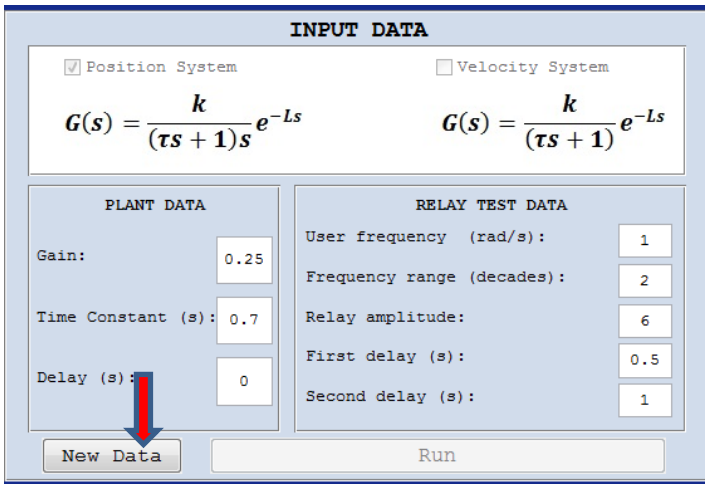


Figura 31: New data button.