MANUAL DE LA PLATAFORMA PARA OBTENER LA RESPUESTA EN FRECUENCIA

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

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

MANUAL DE LA PLATAFORMA PARA OBTENER LA RESPUESTA EN FRECUENCIA

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

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

DIRECTOR DEL PROYECTO:

HECTOR RAMIRO PEREZ RODRIGUEZ
Ingeniero

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

PAG

LISTA DE FIGURAS

OBJETIVO

El propósito del proyecto es identificar un sistema a través de la respuesta en frecuencia, por lo que se creo un espacio de trabajo con la GUI de matlab. Desde allí se enviara y recibirá datos, asimismo se podrá dibujar el diagrama de bode.

1. RESPUESTA DEL SISTEMA A UNA ENTRADA ESCALON

Figura 1. Respuesta a un step

En el Botón de *respuesta a* un step se envía un escalón por el puerto digital hacia el motor, y se adquiere la respuesta del motor al STEP, así se sabe cuál es el tiempo de establecimiento del motor. La respuesta del motor es pasada por un filtro SMOOTH, con el objetivo de suavizar la señal. Con la señal suavizada, se encuentra el valor del tiempo de establecimiento y es guardado en una variable llamada xm, con el fin de saber en qué momento la respuesta del sistema es estable, y poder analizarla a partir de ese punto. El valor de xm se calcula mediante interpolación lineal.

1.1 Configuración de salida digital y entrada análoga

```
DIO=digitalio('nidaq','Dev3')
linea=addline(DIO,0,'out')
ai = analoginput('nidaq','Dev3')
addchannel(ai,0)
```

Para la configuración digital, como primer paso se crea el objeto, ya que estos son los elementos básicos de la Toolbox utilizados para acceder al dispositivo. Seguidamente se coloca el adaptador que en este caso es nidaq y se coloca el respectivo ID que esta asociado con el hardware de la tarjeta. Después de crear un objeto, debe agregársele líneas. Las líneas son los elementos básicos con los cuales se adquiere o entrega datos.

Para agregar una línea al objeto digital se usa la función ADDLINE.

Esta función requiere que se le especifique el nombre del objeto, el número (ID) de la línea del hardware y su dirección (ya sea entrada o salida).

```
addline(Objeto,IDLíneas, 'Dirección')
```

Para la configuración de la entrada análoga, se debe hacer el mismo proceso. Primero se crea el objeto conociendo el adaptador y el ID, luego se agregan las líneas por la que se va a recibir los datos, esto se hace con addchannel.

Chan = addchannel(Objeto,IDcanales)

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

IDcanales: IDs de los canales del hardware, son valores numéricos asignados por el fabricante para identificar los canales de sus dispositivos, para los de National Instruments empiezan desde 0 hasta el número total de canales de salida análoga.

1.2 Modo de entrada análoga y configuración de tasa de muestreo

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

set(ai,'samplerate',333);
```

Existen dos formas de medir las entradas análogas, esto depende si la señal de entrada es referenciada a tierra o es una señal flotante. Como entrada de nodo simple (monopolar) y como entrada diferencial. El modo en este caso es el modo diferencial, debido a que la señal esta referenciada a tierra.

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

El valor de InputType determina el número de canales del hardware que pueden agregarse a un objeto. Si se toma el valor por defecto (Differential), el máximo número de canales que puede contener el objeto es 4, mientras que al establecer el valor de InputType para entradas de simple nodo (SimpleEnded) los 8 canales de entrada análoga que tiene la tarjeta pueden ser agregados al objeto.
Para configurar la tasa de muestreo de la entrada análoga, primero se debe conocer que la máxima tasa de muestro es de 10khz.

Se tiene la desventaja de que si se piensa usar un canal como entrada análoga se podrá adquirir a 10khz, pero si se usa 2 canales, la frecuencia de muestro se divide en los dos canales. Esto se debe a que la tarjeta posee un multiplexor que comparte el amplificador y el ADC con todos los canales, por lo que limita la velocidad de adquisición. Para poder definir la tasa de muestreo se utiliza samplerate.

```
set(ai,'samplerate',333);
```

Se define la tasa de muestreo de 333, debido a que el tiempo de adquisición dura 3 segundos, la señal tendrá 999 muestras.

1.3 Envio de datos digitales

```
n=10;
putvalue(DIO,0)
pause(n)
putvalue(DIO,1)
```

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

```
putvalue(Objeto,Datos)
```

Objeto: Nombre del objeto digital creado y al cual se le agregó la línea como salida.
Datos: Números decimales o vectores binarios, el cual el 0 representa un bajo, y el 1 representa 5v, un alto.

1.4 Extraer datos analogos

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

Para poder empezar a extraer datos análogos primero se debe abrir el puerto de entrada análoga (start), si esto no se hace la extracción de datos no se logra. Para poder extraer datos se utiliza la función getdata.

1.5 Filtro

```
b=ones(1,10)/10;
yt1=filtfilt(b,1,mode2);
cc=smooth(yt1,0.3);
fu=length(cc);
dd=cc(950)
t1=0:3/999:3;
t=0:10/1499:10;
axes(handles.axes5)
plot(t1,cc)
zoom on
grid
```

Primero se utiliza un filtro digital con el propósito de reducir al mínimo la puesta en marcha y terminar transitorios, para así coincidir las condiciones iniciales. Inicialmente se filtra el vector x, su respuesta se rota y aplica nuevamente el mismo filtro. La respuesta final evita la distorsión de fase propia de los filtros IIR.

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

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

Consigo a esto, se aplica un filtro llamado SMOOTH, el cual suaviza la señal y el resultado lo guarda en un vector. El procedimiento lo hace usando filtrado de media móvil. La duración de la media móvil es variada con el SPAN. Por defecto es 5.

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

La media móvil es un filtro digital para una tarea común: la reducción de ruido aleatorio, manteniendo una fuerte respuesta al escalón. El filtro de media móvil opera un promedio de un número de puntos de la señal de entrada para producir cada punto de la señal de salida.

No sólo es el filtro de media móvil muy bueno para muchas aplicaciones, es óptimo para un problema común, la reducción de ruido blanco al azar mientras se mantiene la respuesta al escalón más aguda.

1.6 Tiempo de establecimiento

El tiempo de establecimiento, es el tiempo que se gasta el sistema en estabilizarse. Este se encuentra conociendo el T que es el 63% de la señal, y al multiplicar este valor por 4 se encuentra el tiempo de establecimiento. La localidad de este tiempo es guardado en xm. Para encontrar el valor más cercano del tiempo se realiza una interpolación lineal.

```
%vamos a buscar el tiempo de establecimiento
wwx=dd*0.63
% wwx=4.345*0.63;
for i=1:1:length(cc)
   if cc(i)<wwx
      m=cc(i)
        else
   end
end
for ii=1:1:length(cc)
   if cc(ii)==m
      xm2=ii
   else
   end
end
e=xm2+1
n=cc(e)
m5=m
rt1=t1(xm2)% punto1(rt1,m5)
rt2=t1(e)%  punto2(rt2,n)
```

En este punto se aplica la interpolación lineal, con el propósito de poder encontrar el punto más cercano a xy, que viene a ser T. Al mismo tiempo se va visualizando el tm y el T.

```
m1=(n-m5)/(rt2-rt1)
b=m5-(m1*rt1)
xy=(wwx-b)/m1
set(handles.text6,'string',xy);
TS2=xy*4
set(handles.text7,'string',TS2);
```

Después de haber encontrado el valor de TS2, se busca en el vector tiempo para poder encontrar la posición, con el objetivo de encontrar de qué punto a qué punto se debe cortar las señales sinodales.

```
for i2=1:1:length(t)    %cambio el er, dependiendo del tamaño del vector
   if t(i2)<TS2
      xm=i2;
 else
   end
end
delete(DIO)
clear DIO
delete(ai)
clear ai
```

2. IDENTIFICACION EN LAZO ABIERTO

Figura 2. Identificación en lazo abierto

Al encontrar el tiempo de establecimiento del sistema, el siguiente paso es la identificación del mismo, a través de algunos métodos conocidos, teniendo en cuenta que algunos de ellos tienen menor error que otros.

En este recuadro se escoge el método. Con la ayuda del Listbox se toma el método que se desee y se cargan las constantes de cada uno, luego se identifica el sistema encontrando las variables tm y T. Teniendo las 2 constantes se armar la función de transferencia. Luego se le aplica un escalón y se compara con la respuesta al STEP anterior.

2.1 Modelos de primer orden más tiempo muerto

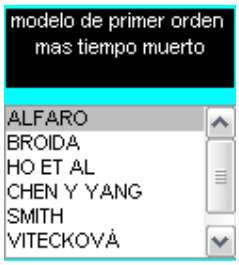

Figura 3. Metodos de primer orden mas tiempo muerto

Con el Listbox se escoje el metodo de identificacion que se desea utilizar. Para cada uno se cargan unos constantes, donde a y b son constantes con valores fijos, y ct1 y ct2 son los porcentajes de los 2 puntos que se toman de la señal, con estos 4 valores se encuentra el T y el tm.

```
if fun4==1
    a=0.91;
    b=1.26;
    ct1=0.25;
    ct2=0.75;
elseif fun4==2
    a=0.55;
    b=2.80;
    ct1=0.28;
    ct2=0.40;
elseif fun4==3
    a=0.67;
    b=1.29;
    ct1=0.35;
    ct2=0.85;
elseif fun4==4
    a=0.14;
    b=1.54;
    ct1=0.33;
    ct2=0.67;
    elseif fun4==5
    a=0.15;
    b=1.50;
    ct1=0.28;
    ct2=0.63;
elseif fun4==6
    a=1.24;
    b=1.49;
    ct1=0.33;
    ct2=0.7;
       end
```

2.2 Identificacion del sistema

Figura 4. Identificar sistema

Este Button identifica el sistema teniendo en cuenta las constantes para cada metodo.

Cargando algunas variables, se encuentra el tiempo de los 2 puntos que se adquieren, siempre utilizando el método de 2 puntos.

```
deltau=5;
yu=dd;%%esto viene
kp=yu/deltau;
t1m=yu*ct1
t2m=yu*ct2
t=0:3/999:3;
```

A continuación, se aplica interpolación lineal con la finalidad de encontrar el valor mas aproximado del tiempo cuando la respuesta del sistema alcanza el valor de t1m. Este es guardado en xy2.

```
for i=1:1:length(cc)
    if cc(i)<t1m
     m(i)=cc(i)
   else
    end
end
for t11=1:1:length(m)
    if m(length(m))>m(length(m)-1)
    m5=m(length(m));
   end
end
e=t11+1
n=cc(e)
rt1=t(t11)% punto1(rt1,m5)
rt2=t(e)%  punto2(rt2,n)
m1=(n-m5)/(rt2-rt1)
b1=m5-(m1*rt1)
```

```
xy2=(t1m-b1)/m1
```

A continuación, se aplica interpolación lineal con la finalidad de encontrar el valor más aproximado del tiempo cuando la respuesta del sistema alcanza el valor de t2m. Este es guardado en xy3.

```
for i1=1:1:length(cc)
   if cc(i1)<t2m
      mm(i1)=cc(i1)
   else
   end
end
for t12=1:1:length(mm)
   if mm(length(mm))>mm(length(mm)-1)
   m55=mm(length(mm));
      end
end
e1=t12+1
n1=cc(e1)
rt11=t(t12)% punto1(rt11,m55)
rt22=t(e1)%  punto2(rt22,n1)

m11=(n1-m55)/(rt22-rt11)
b2=m55-(m11*rt11)
xy3=(t2m-b2)/m11
```

En el siguiente paso se encuentra el T, el polo, el tiempo muerto, el tiempo de establecimiento (TS2). Además se compara el valor del tiempo de establecimiento, con el del sistema real.

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

2.3 Respuesta a un escalón

Figura 5. Boton Step

Aquí se construye con el T y el tm, la funcion de transferencia, se aplica a la entrada un STEP, buscando comparar lo teorico con lo experiemntal, observando que son muy similares.

El valor de num y den, son ya conocidos, el problema es como convertir el tm en numerador y denominador. Con la funcion PADE se encuentra la solucion a este problema. Finalmente se multiplican las 2 ecuaciones, y se le aplica un STEP.

```
num=kp;
den=[T 1];
[num1,den1]=pade(tm,1);
x1=tf(num,den)
x2=tf(num1,den1);
x3=x1*x2
nu=num*num1;
 xde=conv([den],[den1]);
axes(handles.axes4)
step(x3)
grid
zoom on
```

3. FRECUENCIAS A TRABAJAR

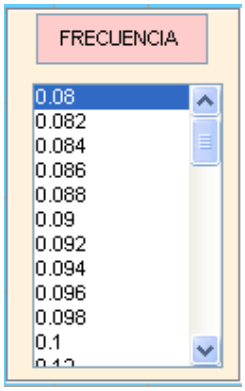

Figura 6. Selección de frecuencias

Pimero se escojen las frecuencias a trabajar. El paso entre frecuencia y frecuencia se hace bien pequeño, con la finalidad de construir el diagrama de bode de la manera mas aproximada. Para cada frecuencia se carga la constante r, que representa la cantidad de periodos que posee la señal. Y *a* es un contador.

```
if fun4==1
        r=0.8;
        a=1;
elseif fun4==2
        r=0.82;
        a=2;
elseif fun4==3
        r=0.84;
        a=3;
elseif fun4==4
        r=0.86;
        a=4;
elseif fun4==5
        r=0.88;
        a=5;
elseif fun4==6
        r=0.9;
        a=6;
elseif fun4==7
        r=0.92;
        a=7;
elseif fun4==8
```

```
  r=0.94;
    a=8;
elseif fun4==9
    r=0.96;
    a=9;
elseif fun4==10
    r=0.98;
    a=10;
elseif fun4==11
    r=1;
    a=11;
 elseif fun4==12
    r=1.2;
    a=12;
elseif fun4==13
    r=1.4;
    a=13;
elseif fun4==14
    r=1.6;
    a=14
elseif fun4==15
    r=1.8;
    a=15;
elseif fun4==16
    r=2;
    a=16;
elseif fun4==17
    r=2.2;
    a=17;
elseif fun4==18
    r=2.4;
    a=18;
elseif fun4==19
    r=2.6;
    a=19;
elseif fun4==20
    r=2.8;%problem
    a=20;
elseif fun4==21
    r=3;
```

```
    a=21;
  elseif fun4==22
    r=3.2;
    a=22;
  elseif fun4==23
    r=3.4;
    a=23;
  elseif fun4==24
    r=3.6;
    a=24;
  elseif fun4==25
    r=3.8;
    a=25;
  elseif fun4==26
    r=4;
    a=26;
elseif fun4==27
    r=4.2;
    a=27;
  elseif fun4==28
    r=4.4;
    a=28;
  elseif fun4==29
    r=4.6;
    a=29;
  elseif fun4==30
    r=4.8;
    a=30;
  elseif fun4==31
    r=5;
    a=31;
  elseif fun4==32
    r=5.5;
    a=32;
  elseif fun4==33
    r=6;
    a=32;
    elseif fun4==34
    r=6.5;
    a=32;
```

```
    elseif fun4==35
    r=7;
    a=32;
    elseif fun4==36
    r=8;
    a=32;
    elseif fun4==37
    r=9;
    a=32;
    elseif fun4==38
    r=10;
    a=32;
    elseif fun4==39
    r=15;
    a=32;
   elseif fun4==40
    r=20;
    a=32;
    elseif fun4==41
    r=25;
    a=32;
    elseif fun4==42
    r=30;
    a=32;
    elseif fun4==43
    r=35;
    a=32;
    elseif fun4==44
    r=40;
    a=32;
    elseif fun4==45
    r=50;
    a=32;
    elseif fun4==46
    r=60;
    a=32;
    elseif fun4==47
    r=70;
    a=32;
    elseif fun4==48
```

```
    r=80;
    a=32;
    elseif fun4==49
    r=90;
    a=32;
    elseif fun4==50
    r=100;
    a=32;
end
```

4. SIMULACION DEL SISTEMA A ENTRADAS SENOIDALES

Antes de enviar señales senoidales hacia el motor, primero se simula el sistema identificado a entradas senoidales para observar como se comportan con respecto a las señales enviadas y señales recibidas de la planta, que en este caso es un motor dc. Se observa que la señal de salida, a medida que se aumenta la frecuencia, disminuye su amplitud y el desfase va aumentando.

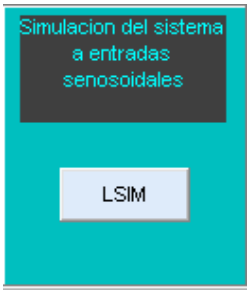

Figura 7. Botón de simulación de sistemas a entradas sinusoidales

```
rr=10/r;
frec(a)=1/(rr)
ad=frec(a);
t=(0:10/1499:10);
u=2.5*sin(2*pi*ad*t);
```

Esta es la función que simula el sistema a entradas senoidales. Esta función guarda la señal de salida en y.

```
[y,x]=lsim(nu,xde,u,t);
t2=t(xm:1500);
y2=y(xm:1500);
u2=u(xm:1500);
xc=max(y2);
xf=min(y2);
xf2=xc-xf;
xg=xf2/2;
axes(handles.axes3)
plot(t2,u2)
zoom on
grid
axes(handles.axes4)
plot(t2,y2)
zoom on
grid
t233=y2;
```

5. ENVIO DE LA SEÑAL ANALOGA

Figura 8. Botòn envio de datos

Teniendo la frecuencia seleccionada, el siguiente paso es enviar la señal hacia el motor, por lo que el Botón de Entrada se encarga de enviar los datos, además, a través de una rutina se encuentra el tiempo de cruce por cero por primera vez, en este caso 2,5, debido a que la señal es enviada en un rango de 0 a 5v. El valor de ese tiempo es guardado en x. También se encuentra la frecuencia de la señal, la cual es guardada en frec.

5.1 Configuracion para enviar una señal análoga

```
ao = analogoutput('nidaq','Dev1');
```

```
addchannel(ao,0);
ai = analoginput('nidaq','Dev1');
addchannel(ai,0);
```

Para la configuración de salida análoga primero se crea el objeto conociendo el adaptador y el ID, luego se agregan las líneas por la que se va a recibir los datos, esto se hace con addchannel.

Chan = addchannel(Objeto,IDcanales)

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

IDcanales: IDs de los canales del hardware, son valores numéricos asignados por el fabricante para identificar los canales de sus dispositivos, para los de National Instruments empiezan desde 0 hasta el número total de canales de salida análoga.

5.2 Modo de entrada análoga y configuración de tasa de muestreo

```
set(ai,'inputType','singleEnded')
set(ai,'samplerate',300);
```

Existen dos formas de medir las entradas análogas, dependiendo si la señal de entrada es referenciada a tierra o es una señal flotante. Como entrada de nodo simple (monopolar) y como entrada diferencial. El modo a utilizar es de entrada diferencial, debido a que la señal esta referenciada a tierra.

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

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

Para configurar la tasa de muestreo de la entrada análoga, primero se debe conocer que la máxima tasa de muestro es de 10khz. Se tiene la desventaja de que si se piensa usar un canal como entrada análoga se podrá adquirir a 10khz, pero si se usa 2 canales, la frecuencia de muestro se divide en los dos canales. Esto se debe a que la tarjeta posee un multiplexor que comparte el amplificador y el ADC con todos los canales, por lo que limita la velocidad de adquisición. Para poder hacer esto se utiliza samplerate.

```
set(ai,'samplerate',300);
```

Se define la tasa de muestreo de 300, debido a que el tiempo de adquisición dura 10 segundos, la señal tendrá 3000 muestras. Se puede colocar un valor mayor, la tasa de muestreo máxima es de 10khz.

5.3 Tasa de muestreo de salida análoga

```
d=10;
set(ao,'samplerate',150)
actualrate=get(ao,'samplerate');
```

La tasa de muestro de salida análoga es una limitante debido a que el máximo valor es de 150hz.

5.4 Envio y extracción de datos análogos

La señal senoidal es construida con la función linspace, donde lens es el número de muestras a enviar.
El putsample, se utiliza para enviar datos análogos hacia el puerto de salida análogo, y el getsample se encarga de extraer dato a dato del puerto de entrada análogo.

```
lens=actualrate*d;
pasos=lens-1;
datos =2.5*sin(linspace(0,2*pi*r,lens))+2.5;
t=(0:d/pasos:d)
```

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

5.5 Corte de señales

Se tiene como finalidad poder tomar la señal desde el momento en que se encuentra en estado estable.

```
tt2=t(xm:1500);% recorte de tiempo, desde t11 que es el ts
er=(1500-xm)+1;
t23=datos(xm:1500)%recorte de datos, desde t11 que es ts
axes(handles.axes1)
 plot(tt2,t23)
zoom on
grid
```

5.6 Tiempo de cruce por cero

Se proponer una forma de poder encontrar los tiempos de cruces cuando cortan la mitad de la señal, y guardarlos en una variable x. Para encontrar un valor aproximado se realiza nuevamente una interpolación lineal.

```
for i2=1:1:length(t23)     %cambio el er, dependiendo del tamaño del vector
   if t23(i2)<2.5   %cambio el 2.5..por el valor del q deseo buscar
      m(i2)=t23(i2);
       else
   end
end
if m(1)==0
   for tt3=1:1:length(m)
      if m(tt3)>0
```

```
            m5=m(tt3); %abajo
            e=tt3-1;
            n=t23(e);%arriba
            re1=tt2(e)%  punto1(re1,n)
            re2=tt2(tt3)% punto2(re2,m5)
         break
        else
       end
    end
else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1)
            e=tt3;
            n=t23(e);%arriba
            re1=tt2(e-1)% punto1(re1,m5)
            re2=tt2(tt3)%  punto2(re2,n)
            break
        else
       end
    end
end
m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x=(2.5-b)/m1
```

5.7 Magnitud

Se guardan las frecuencias de las señales, asimismo se va calculando la amplitud de la señal de la salida análoga y esta se visualiza.

```
rr=10/r;
frec(a)=1/(rr)
ac=frec(a);
```

```
m7(a)=x
 xc=max(datos);
xd=min(datos);
xe=(xc-xd)/2;
set(handles.text2,'string',xe);
```

6 ADQUIRIR SEÑAL ANALOGA

Figura 9. Botòn de adquisiciòn de señal analoga

En el botón Salida, se adquiere la señal que envía el motor y se filtra la misma, debido a que tiene ruido. El filtro que se utiliza es el SMOOT, con el objetivo de poder suavizar la señal. Asimismo se encuentra el pico por encima y por debajo de la señal, para así poder encontrar su amplitud.

6.1. Filtro

La señal que es adquirida es filtrada. Luego es recortada donde ya se encuentra en estado estable.

```
cc=smooth(mode,'moving');


tt2=t(xm:1500);% recorte de tiempo, desde t11 que es el ts
er=(1500-xm)+1;
t23=cc(xm:1500);%recorte de datos, desde t11 que es ts..yt va a ser cc
```

6.2 Amplitud

Para poder encontrar la amplitud, primero se encuentra el primer pico por encima y el primer pico por debajo, los restamos y los dividimos en 2, y así se encuentra la magnitud.

6.2.1 Pico por debajo

Se recorre la señal, se guarda en un vector todos los valores que componen los picos que están por debajo, se recorta los que componen al primer pico, y luego se pregunta cuál es el menor de todos, y ese va a ser el pico por debajo.

```
            for i2=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
                    if t68(i2)<0   %cambio el 2.5..por el valor del q deseo buscar
                                                m(i2)=t68(i2);
   else
   end
end
 if m(1)==0
   for tt3=1:1:length(m)
      if m(tt3)<0
         m5=m(tt3); %abajo
         e=tt3;
         n=t68(e);%arriba
         re1=tt2(e);%  punto1(re1,n)
         re2=tt2(tt3);% punto2(re2,m5)
         break
      else
      end
   end
else
   for tt3=1:1:length(m)
      if m(tt3)==0
         m5=m(tt3-1);
         e=tt3-1;
         n=t68(e);%arriba
         re1=tt2(e-1);% punto1(re1,m5)
         re2=tt2(tt3);%  punto2(re2,n)
         break
      else
        end
   end
end
if m(1)==0
mm=m(tt3:length(m));
for ttt3=1:1:length(mm)
```

```
   if mm(ttt3)<0
      m55=mm(ttt3);
      e1=(ttt3-1)+tt3;
%           nn=y(e1)%arriba
%           re11=t2(e1)%  punto1(re1,n)
         re22=tt2(ttt3);% punto2(re2,m5)
         ty2=tt3;
      pn=m(tt3:e1);
   else
      break
   end
end
else
   mm2=m(1:e)
   pt=mm2(1);
   ty2=1;
pn=m(1:e);
end
 picopordebajo=min(pn)%amplitud de la señal
```

6.2.2 Pico por encima

Se hace el mismo procedimiento anterior, se recorre la señal, se guarda en un vector todos los valores que componen los picos que están por encima, se recorta los que componen al primer pico, luego se pregunta cual es el mayor, y ese seria el pico por encima.

```
for i22=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
   if t68(i22)>0   %cambio el 2.5..por el valor del q deseo buscar
      m(i22)=t68(i22);
   else
   end
end
if m(1)==0
   for tt3=1:1:length(m)
      if m(tt3)>0
         m5=m(tt3); %abajo
```

```
            e=tt3;
            re1=tt2(tt3-1);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)
            break
        else
        end
    end
else
    for tt3=1:1:length(m)
        if m(tt3)==0
            m5=m(tt3-1);
            e=tt3-1;
            re1=tt2(tt3-1)% punto1(re1,m5)
            re2=tt2(tt3)%  punto2(re2,n)
            break
        else
        end
    end
end
if m(1)==0
mm=m(tt3:length(m))
for ttt3=1:1:length(mm)
    if mm(ttt3)>0
        m55=mm(ttt3);
        e1=ttt3-1;
%            nn=y(e1)%arriba
          %re11=t2(e1-1)%  punto1(re1,n)
          %re22=t2(ttt3)% punto2(re2,m5)
          e11=tt3+e1;
          pn=m(tt3:e11);
          ty2=tt3;
    else
        break
    end
end
else
    mm2=m(1:length(m));
pt=mm2(1);
pn=m(1:e);
ty2=1;
```

```
end
picoporencima=max(pn)%
```

Luego de tener el valor de los 2 picos se procede a encontrar la amplitud de la señal que se adquirió.

```
xgg2=picoporencima-picopordebajo
xgg=(xgg2)/2;
axes(handles.axes2)
plot(tt2,xgg)
zoom on
grid
```

Por último se guarda la magnitud del sistema

```
A(a)=20*log10(xgg/xe)
```

7. TRANSFORMADA DE FOURIER

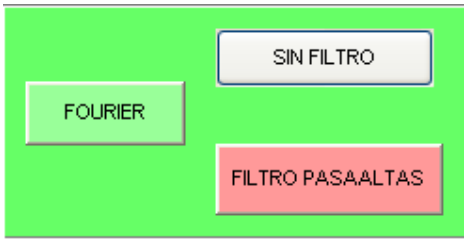

Figura 10. Aplicaciones para la señal adquirida

Como finalidad se tiene encontrar el espectro de frecuencia de la señal, con el proposito de que el usuario pueda observar, que a medida que la frecuencia va aumentando, se vizualiza en la señal una baja frecuencia, por tal motivo, se aplica un filtro pasa altas para poder elimar las frecuencias que no son necesarias ni utiles.

7.1 Fourier

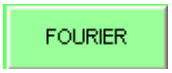

Figura 11. Boton para calculo de FFT

Se dibuja el espectro de frecuencias de la señal, para que el usuario pueda ver que frecuencias que la componen.

```
L=length(t68);
Fs=150;
 NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(t68,NFFT)/L;
fp = Fs/2*linspace(0,1,NFFT/2+1);
 % Plot single-sided amplitude spectrum.
aa=abs(Y(1:NFFT/2+1));
axes(handles.axes6)
plot(fp,2*aa)
zoom on
 grid
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
xc=max(2*aa)
xd=2*aa;
```

El objetivo de este bucle es el de poder encontrar el valor de la frecuencia de la señal en el espectro de frecuencia y visualizarla para que el usuario pueda interactuar con ella

.

```
for i2=1:1:length(xd)
   if xd(i2)==xc   %cambio el ..por el valor del q deseo buscar
      m=xd(i2)
   else
   end
end
frecuenciadelasenal=xd
frecc(a)=xd
set(handles.text8,'string',frecuenciadelasenal);
```

7.2. Sin filtro

Figura 12. Botón para no filtrar la señal adquirida

No a todas las frecuencias se debe aplicarle un filtro, el usuario tiene la posibilidad de filtrar o no la señal. Este Button contiene variables importantes para dibujar el diagrama de bode. Las variables son el tiempo de retardo entre las 2 señales y la amplitud de la señal recibida. Ademas se encuentran los tiempos de cruce por cero.

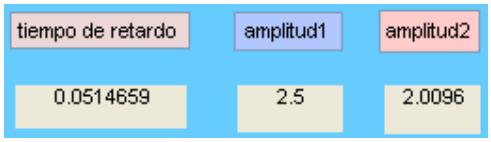

Figura 13. Variables a obtener

Se identifica la señal cuando está por debajo de 0, con el objetivo de saber cuando está por encima. Haciendo esto se conoce el primer cruce por 0.

```
for i2=1:1:length(t233)     %cambio el er, dependiendo del tamaño del vector
   if t233(i2)<0   %cambio el 2.5..por el valor del q deseo buscar
      m(i2)=t233(i2);
   else
       m(i2)=0;
   end
end
if m(1)==0
   for tt3=1:1:length(m)
      if m(tt3)<0
         m5=m(tt3) %abajo
         e=tt3-1;
         n=t233(e);%arriba
         re1=tt2(e);%  punto1(re1,n)
         re2=tt2(tt3);% punto2(re2,m5)
         break
      else
```

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

Inmediatamente se recorta la señal apenas surja un cambio en el vector m, queriendo decir que paso por el primer cruce y así poder identificar el segundo cruce por cero. De este modo se puede conocer el tiempo de retardo. El valor del retardo se guarda en x4.

```
m2=m(tt3:length(m));
ty2=tt2(tt3:length(m));
if a>11
  if m2(1)==0
    for ttr3=1:1:length(m2)
      if m2(ttr3)<0
         m56=m2(ttr3) %abajo
         e6=ttr3-1;
         tty1=ty2(ttr3)
         tty2=ty2(e6)
         break
      else
      end
   end
else
   for ttr3=1:1:length(m2)
     if m2(ttr3)==0
         m56=m2(ttr3-1)
```

```
            e6=ttr3;
            tty1=ty2(ttr3-1)
            tty2=ty2(e6)
            break
        else
        end
    end
 end
for i24=1:1:length(tt2)
    if tt2(i24)==tty1
        mt1=i24;
         break
    else
    end
end
for i25=1:1:length(tt2)
    if tt2(i25)==tty2
        mt2=i25;
         break
    else
    end
end
m561=t233(i24);
m562=t233(i25);
 m11=(m562-m561)/(tty2-tty1);
b=m561-(m11*tty1);
x2=(0-b)/m11
m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x3=(0-b)/m1
if x3>x
    x4=x3-x;
else
    x4=x2-x;
end
 else
m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x3=(0-b)/m1
x4=x3-x;
```

```
end
set(handles.text1,'string',x4);
```

Se visualiza la amplitud de la señal que es adquirida.

```
amplitud(a)=xg
set(handles.text3,'string',xg);
```

Se guarda en un vector la fase del sistema.

```
w(a)=2*pi*ac
we(a)=-ac*x4
wr=we(a)
wx(a)=(wr*180)/pi
```

7.3. Filtro pasa altas

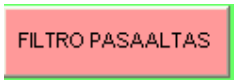

Figura 14. Botòn de aplicación de filtro pasaaltas

Este BUTTON fue creado para el caso donde la señal este montada sobre otra otra señal pero de baja frecuencia. Aquí igualmente se encuentra el tiempo de retardo, y la amplitud de la señal filtrada, y ademas se visualizan en el mismo lado donde lo muestra el BUTTON sin filtro. Tambien va a visualizar la frecuencia que se elimino al aplicarel filtro.

Primero que todo le aplicamos el filtro pasa altas, eliminando lo que esta a la izquierda de la frecuencia de la señal por debajo de 0.15, valor que se escogió por diseño. Luego de encontrar el numerador y el denominador de el filtro, se procede a aplicarle un filtro de suavizado.

```
ad=ac-0.15;
fn=ad/ac;
fc=fn/(ac/2);
[b,c]=butter(1,fc,'high');
```

```
yt=filtfilt(b,c,cc);
cc3=smooth(yt);
 cc2=smooth(cc3,'moving');
cc4=smooth(cc2,0.01);
t26=cc4(xm:1500);
t68=cc4-median(cc4);
```

Se encuentra el pico por debajo y por encima de la señal, para poder encontrar su amplitud después de haber sido filtrada.

```
for i2=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
   if t68(i2)<0   %cambio el 2.5..por el valor del q deseo buscar
      m(i2)=t68(i2);
   else
   end
end
Pico por debajo
if m(1)==0
   for tt3=1:1:length(m)
      if m(tt3)<0
         m5=m(tt3); %abajo
         e=tt3;
         n=t68(e);%arriba
         re1=tt2(e);%  punto1(re1,n)
         re2=tt2(tt3);% punto2(re2,m5)
         break
      else
      end
   end
else
   for tt3=1:1:length(m)
      if m(tt3)==0
         m5=m(tt3-1)
         e=tt3-1;
         n=t68(e);%arriba
%           re1=tt2(e-1);% punto1(re1,m5)
%           re2=tt2(tt3);%  punto2(re2,n)
         break
      else
```

```
        end
    end
end
if m(1)==0
mm=m(tt3:length(m));
for ttt3=1:1:length(mm)

    if mm(ttt3)<0
        m55=mm(ttt3);
        e1=(ttt3-1)+tt3;
%           nn=y(e1)%arriba
%           re11=t2(e1)%  punto1(re1,n)
%           re22=tt2(ttt3);% punto2(re2,m5)
          ty2=tt3;
        pn=m(tt3:e1);
    else
        break
    end
end
else
    mm2=m(1:e)
    pt=mm2(1);
    ty2=1;
pn=m(1:e);
end
picopordebajo=min(pn)%amplitud de la señal
n4=picopordebajo*-1
for ty=1:1:length(pn)
    if pn(ty)==picopordebajo
        ty1=ty;
    else
    end
end
tye=ty1-1;
tye1=ty2+tye;
tiem=tt2(tye1);%tiempo de la amplitud de la señal
```

Igualmente se encuentra el pico por encima.

```
for i22=1:1:length(t68);     %cambio el er, dependiendo del tamaño del vector
```

```
   if t68(i22)>0   %cambio el 2.5..por el valor del q deseo buscar
      m(i22)=t68(i22);
   else
   end
end
 if m(1)==0
   for tt3=1:1:length(m)
      if m(tt3)>0
         m5=m(tt3); %abajo
         e=tt3;
         re1=tt2(tt3-1)%  punto1(re1,n)
         re2=tt2(tt3)% punto2(re2,m5)
         break
      else
      end
   end
else
   for tt3=1:1:length(m)
      if m(tt3)==0
         m5=m(tt3-1)
         e=tt3-1;
         re1=tt2(tt3-1)% punto1(re1,m5)
         re2=tt2(tt3)%  punto2(re2,n)
         break
      else
      end
   end
end
if m(1)==0
mm=m(tt3:length(m))
for ttt3=1:1:length(mm)
   if mm(ttt3)>0
      m55=mm(ttt3)
      e1=ttt3-1;
%           nn=y(e1)%arriba
         %re11=t2(e1-1)%  punto1(re1,n)
         %re22=t2(ttt3)% punto2(re2,m5)
         e11=tt3+e1;
         pn=m(tt3:e11);
         ty2=tt3;
```

```
    else
        break
    end
end
else
    mm2=m(1:length(m));
pt=mm2(1);
pn=m(1:e);
ty2=1;
end
picoporencima=max(pn)%amplitud de la señal es esto - n3(popu), eso me da aprox 0.2372
xgg2=picoporencima-picopordebajo
xgg=(xgg2)/2;
```

Asimismo, se encuentra el tiempo de cruce por cero de la señal adquirida, siendo anteriormente filtrada. Estos datos se guardan en la variable x3.

```
%%%%%%%%%
for i2=1:1:length(t233)     %cambio el er, dependiendo del tamaño del vector
    if t233(i2)<2.5   %cambio el 2.5..por el valor del q deseo buscar
        m(i2)=t233(i2);
    else
    end
end
if m(1)==0
    for tt3=1:1:length(m)
        if m(tt3)>0
            m5=m(tt3); %abajo
            e=tt3-1;
            n=t233(e);%arriba
            re1=tt2(e);%  punto1(re1,n)
            re2=tt2(tt3);% punto2(re2,m5)
            break
        else
        end
    end
else
    for tt3=1:1:length(m)
        if m(tt3)==0
```

```
        m5=m(tt3-1);
        e=tt3;
        n=t233(e);%arriba
        re1=tt2(e-1);% punto1(re1,m5)
        re2=tt2(tt3);%  punto2(re2,n)
        break
    else
    end
  end
end
m1=(n-m5)/(re2-re1);
b=m5-(m1*re1);
x2=(2.5-b)/m1
x3=x2-x;
```

El siguiente paso es guardar en un vector la fase del sistema y en otro la magnitud, teniendo estos 2 vectores se construye el diagrama de bode.

```
we(a)=-ac*x3;
wr=we(a);
wx(a)=wr*180/pi
A(a)=20*log10(xg/xe)

axes(handles.axes2)
plot(tt2,y2)%%%ojo se cambio
zoom on
grid
```

Se visualiza la amplitud de la señal adquirida.

```
amplitud(a)=xg
set(handles.text3,'string',xg);
```

8. COMPARACION DE BODE

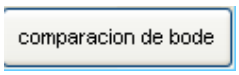

Figura 15. Botòn para graficar el diagrama de bode

Después de cargar las variables de amplitud y el tiempo de retardo entre las 2 señales, se procede a graficar el diagrama de bode.

```
figure
semilogx(w,A)
grid
zoom on
figure
semilogx(w,wx)
grid
zoom on
figure
bode(nu,xde)
grid
zoom on
```

9. PUNTOS CRITICOS PARA PODER ADQUIRIR SEÑALES MEDIANTE MATLAB

- El sistema operativo debe ser xp.

- Se debe conseguir un Matlab que tenga todos los adaptadores necesarios para la adquisición de datos, en este caso se utilizo el R2009.

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

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

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

- La tarjeta 6008 tiene la desventaja de poseer un multiplexor que comparte el amplificador y el ADC con todos los canales, por lo que limita la velocidad de adquisición. Esto quiere decir que si se va a usar un canal como entrada análoga se podrá adquirir máximo a 10khz, pero si se usan 2 canales, la frecuencia de muestro se divide en los dos canales.

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

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

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