Introducción a Matlab y Práctica 1 y 2: Señales Continuas y Discretas

MATLAB® (MATrix LABoratory) es un sistema basado en matrices que permite resolver problemas numéricos relativamente complejos y visualizar los resultados con facilidad, debido a que los planteamientos y las soluciones se expresan de manera similar a su forma matemática original. El objetivo de las simulaciones en el curso de Señales y Sistemas I es utilizar MATLAB® como instrumento para comprender en detalle los aspectos más importantes del curso teórico, además de aprender a utilizar una herramienta de uso extendido en el área de procesamiento de señales y comunicaciones. En general, las señales en MATLAB® son representadas por matrices numéricas, que pueden contener entradas complejas. Todas las variables definidas son matrices. Las matrices con solo una columna o fila son interpretadas como vectores. Todas las matrices representadas en MATLAB® son indexadas comenzando con 1, por ejemplo, *y(1,2)* es el elemento correspondiente a la segunda columna de la primera fila de la matriz *y*. MATLAB® puede ejecutar una secuencia de comandos almacenados en un archivo. Estos archivos se conocen como *archivos .m,* ya que tienen ésta extensión. Los archivos *.m* facilitan mucho del trabajo en MATLAB®, y permiten además la construcción de funciones para realizar tareas específicas. Estos archivos se pueden crear utilizando el editor de texto de MATLAB®, el cual puede ser llamado a través del comando *edit* o utilizando la barra de herramientas en la parte superior de la ventana. Para ejecutar una rutina *.m* basta con escribir en la línea de comando el nombre del archivo sin la extensión (o en el caso de una función el nombre del archivo con los parámetros respectivos), asegurándose de que se encuentra en el directorio apropiado. En la ventana de comandos de MATLAB® es posible el moverse entre directorios con las instrucciones *cd..*, *cd <directorio>:*, etc, similar a los comandos utilizados en DOS. La hoja de trabajo o *workspace* permite escribir instrucciones o secuencias de instrucciones
las cuales se van ejecutando al pulsar <return>.Las variables del *workspace* pueden ser almacenadas en formato *.mat*, utilizando el comando *save,* y pueden ser cargadas de nuevo utilizando el comando *load.* Para obtener más información de éstos y otros comandos puede utilizar la instrucción *help <comando>*.

1. INTRODUCCION AL USO DE MATLAB.

Lo primero que debe comprenderse al usar Matlab es que el manejo de los datos se hace en forma
matricial, las operaciones matemáticas deben estar acorde con este principio. Así un escalar es visto por Matlab como una matriz 1x1, un vector fila de N elementos es una matriz 1xN o si se trata de un vector columna sus dimensiones son entonces Nx1. Empecemos a explorar cada uno de los puntos expuestos haciendo uso del programa. Inicie una sesión en Matlab, para ello basta con hacer doble click sobre el icono del programa, y aparece la interfaz principal que es el *command window*. Usted puede comenzar a trabajar directamente en esta pantalla, que tiene características que le permiten agilizar en cierto grado la escritura de las instrucciones a ejecutar, por ejemplo puede usar las flechas del teclado para volver a escribir comandos que ha usado anteriormente en la misma sesión. La ayuda en línea del programa es bastante completa, puede accesarla directamente desde el menú principal o desde el *command window*. Matlab posee un conjunto de funciones básicas *built in,* cuyo código no es visible al usuario, y funciones más avanzadas basadas en las anteriores, las cuales muchas veces están incluidas en *toolboxes,* clasificados según la aplicación en particular. Cada función tiene una ayuda que Usted puede

visualizar directamente en el *command window* ejecutando el comando ***help* nombre de la función**. Cuando trabaja en el *command window* cada variable creada es almacenada temporalmente en el *workspace*, al finalizar la sesión (ejecutando el comando ***quit*** o ***exit***) las variables se borran, a menos que Usted grabe la sesión, en este caso solo se guardan las variables,
no las instrucciones ejecutadas. Si necesita desarrollar un programa que usará con relativa frecuencia, o que es de una extensión considerable, o sencillamente quiere guardar todos los pasos que siguió en la sesión, lo más conveniente es crear un programa. Para ello abra el *editor/debugger* de programas haciendo click sobre el icono de *New-M File* que esta en el menu principal del *command window*, en este editor puede escribir el programa, correrlo y corregir los errores que se presenten de manera rápida y amena. También puede hacer uso de cualquier editor de texto para copiar sus programas, para que matlab los reconozca solo debe guardarlos con extensión *.m* . Matlab tiene varios tipos de archivo, los archivos con extensión *.mat* son de datos, por ejemplo cuando guarda una sesión, esta se guarda con el nombre que usted le asigne con la extensión *.mat* (ver la ayuda del comando ***save***). Los archivos de programa tienen la extensión *.m*.
Los gráficos tienen extensión *.fig*, aunque el programa le permite guardar las figuras como imágenes (formatos jpg, tiff, etc). Además los archivos de Simulink, la herramienta de programación gráfica de Matlab que aprenderemos posteriormente, se guardan con extensión *.mdl*. Las *funciones* de Matlab son rutinas que devuelven variables de salida dadas ciertas variables de entrada (argumentos de la función). La primera línea de una función debe seguir el siguiente formato:
function [x,y] = *name*(a,b,c)
donde:
x e **y** son las variables de salida ***name*** es el nombre de la función, que se recomienda sea también el nombre con el que se guarda el archivo .m, así en un programa cualquiera se invoca la función escribiendo por ejemplo:
[u,v]=*name*(p,r,s)
Matlab posee prácticamente todas las funciones que se necesitan para hacer procesamiento de señales, cuando necesite realizar un procesamiento y desconozca el nombre de la función correspondiente en Matlab, haga una búsqueda por palabras claves usando el comando ***lookfor* keyword**. Por ejemplo:
***lookfor* plot,**

2. ALGUNOS COMANDOS DE MATLAB.

MATLAB está diseñado para trabajar con matrices. Existen comandos para generar matrices características como ones(matrices llenas de unos), zeros(matrices llenas de ceros), etc. Para trasponer una matriz A se emplea A’ >>A(i, :) accede a la i-ésima fila de la matriz A
>>A (:, j) accede a la j-ésima columna de la matriz A
>>A*B multiplica las matrices A y B
>>X=A/B resuelve X*B=A
>>size Da el tamaño de la matriz
>>length Da la longitud de un vector
>>A(:,[2,4])=A(:, [2,4])*[1 2 3;4 5 6] Las columnas 2 y 4 de A se multiplican por una matriz
RELACIONES
<, >=., <=, ~=(no igual a)
NÚMEROS: Usa números enteros, complejos, reales; Inf es Infinito; i y j representan la raíz

cuadrada de -1
OPERACIONES ARITMÉTICAS: +, -, *. (multiplicación de dos vectores punto a punto), /. (división de vectores punto a punto).
CONDICIONALES
If CONDICION
CONDICION DE VERDADERO
else
CONDICION DE FALSO
end
ITERACIONES
For var=inicio:paso:final
CUERPO
End
ITERACIONES CONDICIONALES
While CONDICION
CUERPO
end
FUNCIONES ESCALARES: Están diseñadas para trabajar con escalares o con matrices pero elemento a elemento: Por ejemplo: sin, cos, log, sqrt (raíz cuadrada),tan, acos,atn,exp, abs…
FUNCIONES VECTORIALES: min, max, sum, mean. Cuando lo hacen sobre matrices calculan a lo largo de las columnas
FIGURAS
>>figure Para abrir una nueva gráfica o figura
>> plot(x,y, estiloelegido) Para graficar y vs x con un trazo definido
Para agregar un título a una figura
>>hold on
>>title(titulo)
Para agregar leyenda en el eje x
>>xlabel(texto)
Para agregar leyenda en eje y
>>ylabel(texto)
MISCELANEOS
>>cd a: cambia directorio a a:
>>dir lista directorio
>>what lista los archivos .m y .mat
>>nombre ejecuta el script nombre
>> A=[1 2 3 ; 4 5 6] crea la siguiente matriz
>>help comando le presentará la ayuda existente para el comando elegido
>>lookfor palabraclave buscará todos aquellos comandos que contienen en su definición la " palabraclave"
>>A=rand(5,4) crea una matriz 5x4 con elementos aleatorios entre 0 y 1
>>a=[1 2 3] crea un vector a
>>B=B(1:2,:) Selecciona de las filas 1 y 2 todas las columnas
>>s=bnnnnnn ….. (Cuando el comando no cabe en una línea se colocan 3 o mas puntos suspensivos y se continua en la otra línea
>>who Permite conocer que variables y matrices están definidas en un momento dado

>>whos Igual a who pero además ofrece todos los detalles de cada matriz
>>clear borra todas las variables
>>clear a Borra solo a
>>eps eps= número mas pequeño representable por Matlab
>>save nombre guarda los arreglos que se han definido en un archivo llamado nombre
>>load nombre carga nombre.mat
>>path es la trayectoria sobre la cual Matlab busca funciones; esto incluye los toolboxes
>>% Indica que de ahí en adelante (misma línea) lo que sigue es comentario

EDICIÓN DE LÍNEAS:

Si se usan las flechitas uno puede ir atrás y reutilizar instrucciones ya escritas

GENERACIÓN DE SECUENCIA

>>X=[inicio:paso:fin]; Colocar ; al final impide que se escriba la secuencia generada (se haga eco en pantalla)

2. INTRODUCCIÓN A LA PRESENTE PRÁCTICA

En esta guía se omitirán las explicaciones del uso de las funciones empleadas en la práctica, use el comando ***help*** cada vez que se encuentre con una nueva función y lea con atención la descripción dada

La introducción de datos en Matlab se puede hacer:

• Cargando un archivo de datos externos (ver el comando ***load***)
• Creando una secuencia de entrada en el editor
• Ejecutando alguna función
• Directamente desde el *workspace*, por ejemplo ejecute las siguientes líneas:

>> A=[1 2 3; 9 8 10 ;1 1 1]

Se crea una matriz A con dimensiones 3x3; observe como se hace la diferenciación entre filas y columnas. Observe que sucede si al final de la instrucción anterior añade el operador**;** La instrucción para crear la matriz A es equivalente, entre otras, a :

>>A=[
>>1 2 3
>>9 8 10
>>1 1 1]

Para obtener algún elemento de A, escriba A(n,m) donde n es la fila y m la columna del elemento deseado, pruebe las siguientes instrucciones:

>>A(2,:)
>>A(1:2,3)
>>*size*(A)
>>*length*(A)
>>y=A.^2
>>t=0:20
>>t=0:0.1:1

Para obtener más información acerca de MATLAB® se recomienda consultar la página web http://www.indiana.edu/~statmath/math/matlab/index.html. Allí puede conseguir muchos otros enlaces de interés.

PRACTICA 1: GENERACIÓN Y GRAFICACIÓN DE SEÑALES CONTINUAS Y DISCRETAS

OBJETIVOS

1. Comprender como se simulan señales continuas y discretas en el tiempo usando MATLAB®
2. Generar señales exponenciales, sinusoidales, cuadrada, diente de sierra y escalón, visualizarlas en forma continua y discreta.
3. Revisar las diferentes modalidades que existen para graficar una señal.

EXPERIMENTO

Genere un archivo .m nuevo. Escriba cada instrucción y ejecútela para ver su funcionamiento. Al terminar la práctica podrá ejecutar todas las instrucciones y mostrar los resultados a su profesor. Se le sugiere separe zonas de ejecución usando la instrucción pause.

SEÑALES CONTINUAS

Antes de obtener una señal continua en el tiempo, primero se debe crear un vector que represente la secuencia temporal, teniendo el cuidado de elegir un espaciamiento entre muestras apropiado. Por ejemplo para generar señales en el intervalo de tiempo , con muestras tomadas cada 0.05s, escriba en la línea de comandos:

>>T=0.05

para definir la separación temporal (en segundos) entre las muestras. Exprese la secuencia temporal que va desde -1 a 1, en pasos T:

>>t=[-1:T:1]

Observe que todos los elementos del vector t fueron mostrados en la pantalla. Para evitarlo, usualmente se coloca un punto y coma (**;**) después de cada instrucción.

Para generar la función real decreciente $x(t)=e^{-t}$, escriba:

>>x=exp(-t);

Dibuje x(t) vs. t:

>>plot(t,x,'-y')

El símbolo '-y' indica las características del trazo: "-" es el tipo de trazo e "y" es el color (en este caso yellow o amarillo). Puede obtener más información de cualquier comando utilice **help**; por ejemplo si Ud. quiere saber mas detalles del comando plot escriba:

>>help plot

Pruebe con las diferentes combinaciones de trazos y colores.

Calcule la exponencial creciente $w(t)=e^{t}$:

>>w=exp(t);

Para graficar w(t) existen tres posibilidades : puede dar el comando

>>clf

para borrar la figura anterior, o puede dibujar directamente en el espacio disponible lo cual borrará la figura que estaba anteriormente. También puede dibujarlas simultáneamente con el comando:

>>hold on

En cualquiera de los tres casos, dibuje después w(t)

>>plot(t,w,':r')

Si desea incluir una cuadrícula en el gráfico escriba, luego de hacer el plot:

>>grid; para eliminarla escriba nuevamente: **>>grid;**

Cada vez que Ud. desee graficar una nueva figura debe usar la instrucción:

>>**figure** o **figure(k)** donde k es el número que será asignado a la figura Calcule y grafique las siguientes funciones con cambios lineales en la escala temporal: $x1(t)=e^{-2t}$ y $x2(t)=e^{-t/2}$. Dibújelas junto a la señal original x(t).

>>x1=exp(-2*t);

>>x2=exp(-t/2);

>>plot(t,x1,'-y',t,x2,'--g')
Observe los siguientes símbolos: '*' para la multiplicación y '/' para la división. Proceda de igual manera para la señal $x3(t) = e^{-2|t|}$. El valor absoluto de *t* se calcula con el comando:
>>abs(t)
Por lo tanto la señal x3 se genera con el siguiente comando:
>>x3=exp(-2*abs(t));
>>plot(t,x3,':m')
Ahora graficaremos varias señales en una misma figura pero en espacios diferentes. Para eso se divide primero la figura en una matriz de subgráficos de las dimensiones que uno desee. Imagine que queremos graficar 4 funciones. Dividimos la figura como una matriz de 2x2 y en cada subgráfico aparecerá una de las señales.
>>subplot(2,2,1); plot(t,x1,'-y');
>>subplot(2,2,2); plot(t,,x2,'--g');
>>subplot(2,2,3); plot(t,x3,'r');
>>subplot(2,2,4); plot(t,x,'-b');
Para generar una señal exponencial compleja $y(t)=e^{j2\pi t}$ escriba en la línea de comandos:
>>y=exp(j*2*pi*t);
Observe que **'j'** y **'pi'** son valores internamente definidos en MATLAB. Corresponden a la unidad imaginaria y al número ð respectivamente. **'i'** también puede emplearse en lugar de **'j'**. Para evitar confusiones se recomienda no usar **'i'** ni **'j'** como variables. La señal **'y'** es compleja, a
diferencia de las señales anteriores. Para comprobarlo escriba:
>>whos
Observe que todas las funciones y valores que se han definido se encuentran disponibles en la memoria. Esto no hace falta si Ud. tiene en la pantalla abierto el workspace. Para observar las partes real e imaginaria de **'y'**, primero cree una nueva figura o espacio de presentación:
>>figure(2)
Luego dibuje las partes real e imaginaria.
>>plot(t,real(y),'-b',t,imag(y),':r')
Las sinusoides reales también pueden ser generadas directamente en MATLAB, por ejemplo si se
quieren generar sinusoides se puede usar sin (para Seno) y cos (para Coseno).
>>v1=sin(pi*t-pi/6);
>>v2=cos(pi*t+pi/4);
Ahora generará una señal cuadrada periódica usando la siguiente instrucción:
>>cuad=square(2*pi*t);
Grafíquela:
>>plot(t,cuad)
Observe que las pendientes no son infinitas. Esto ocurre porque el número de puntos es bajo. Haga una prueba usando mas puntos de tiempo (debe definir otro vector de tiempo y volver a graficar). Revise el help de la función square.
Ahora generará una señal diente de sierra periódica usando la siguiente instrucción:
>>saw=sawtooth(2*pi*t);
Grafíquela:
>>plot(t,saw)
Revise el help de esta instrucción. Para finalizar la práctica generaremos un escalón

>>escalon=[zeros(1,20) ones(1,21)];
>>plot(t,escalon)

SEÑALES DISCRETAS

Se le recomienda hacer esta parte de la práctica en un archivo *.m. Antes de continuar borre todos los valores que se encuentran almacenados en memoria:
>>clear

Esta instrucción también puede emplearse para borrar una sola variable. Por ejemplo:
>>clear w o más de una variable:
>>clear x, v1, v2
Para generar una señal discreta en el tiempo x[n], primero se debe definir un vector índice temporal 'n' apropiado. Por ejemplo, para producir una curva exponencial decreciente x[n]=0.9n en el intervalo escriba:
>>n=[-10:10]
La curva exponencial decreciente x[n] se obtiene escribiendo:
>>x=(0.9).^n;
Donde **'.^ '** representa la operación de elevar **0.9** a cada uno de los elementos de **n**. A continuación grafíquela.
>>stem(n,x)
Obtenga una exponencial creciente:
>>w=(1.11).^n;
Grafíquela:
>>stem(n,w)
Genere y grafique la señal par x3[n]=0.9|n|.
>>x3=(0.9).^abs(n);
>>stem(n,x3);
Calcule y grafique la senoidal compleja y[n]=ejðn/5-ð/3.
>>y=exp(j*pi*n/5-pi/3);
>>stem(n,y);
Grafique las partes real e imaginaria de y[n]. ¿Cuál es el período de la señal?. Justifique su respuesta gráfica y analíticamente. Calcule la función z[n]=x[n]y[n]
>>z=x.*y;
Explique como se interpretan las partes real e imaginaria de z[n].
De modo similar a la parte A, genere dos senoidales reales.
>>v1=cos(pi*n/5-pi/3);
>>v2=sin(pi*n/5+pi/4);
Obtenga las funciones par e impar de cada una.
>>v1par=0.5*(v1+fliplr(v1));
>>v1imp=0.5*(v1-fliplr(v1));
Calcule los valores de las funciones par e impar en n=0
>>v1par(find(n= =0)) %Sin punto y coma al final
>>v1imp(find(n= =0))
Calcule los valores de las funciones par e impar en n=0 para v1, v2, al igual que para las siguientes señales:
>>u=[zeros(1,10) ones(1,11)]; %Escalón unitario discreto
>>e=x.*u;

Para finalizar genere alguna de las señales periódicas que conoció al generar señales continuas, pero en forma discreta. Genere un vector discreto de tiempo N de 200 puntos. Pruebe con square(N/pi). Grafique con stem.

ASIGNACION

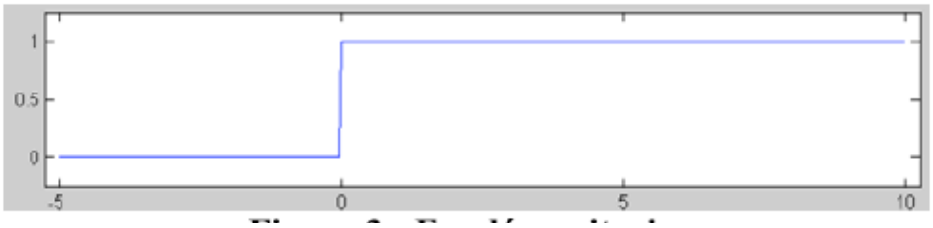
Luego de haber aprendido algunos comandos Ud. debe realizar la siguiente actividad:

1) Genere un vector de tiempo(que se inicie en t = -1) de 20000 puntos en pasos de 1/10000
2) Genere la siguiente señal: -2+3*cos(20*pi*t)+sin(40*pi*t)
3) Genere una señal cuadrada periódica con período igual a 1/10 segundos
4) Genere una señal diente de sierra periódica con período igual a 1/10 segundos
5) Genere una señal igual a sgn(t-0.5)
6) Grafique estas 4 señales en una sola hoja usando subplot y plot; a la última gráfica fíjele un eje de tiempo entre -2 y 2 y un eje de amplitudes entre -2 y 2. A la tercera póngale grilla. A la segunda póngale un título. A la primera póngale nombre a los ejes.
7) Genere un escalón unitario
8) Determine la parte par e impar del escalón
9) Grafique estas 3 funciones una sobre la otra en tres figuras y colores distintos (use stem)
10) Grafíquelas ahora en una misma hoja usando subplot y plot
11) Determine los índices donde la señal diente de sierra toma valores menores a 0.005 y mayores a -0.005

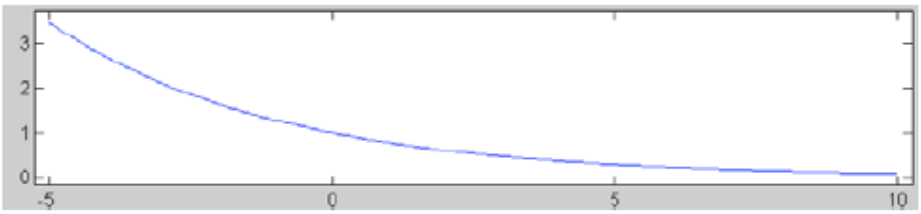
Práctica 2. Formas de ondas

Utilizando lo visto en la práctica #1 obtenga en el Matlab las siguientes gráficas

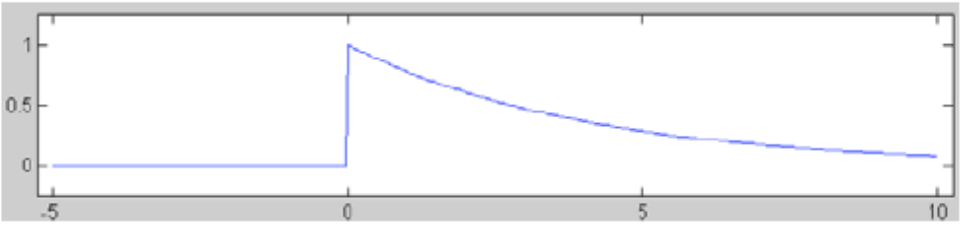
a) $x(t) = u(t)$ (señal "escalón unitario").



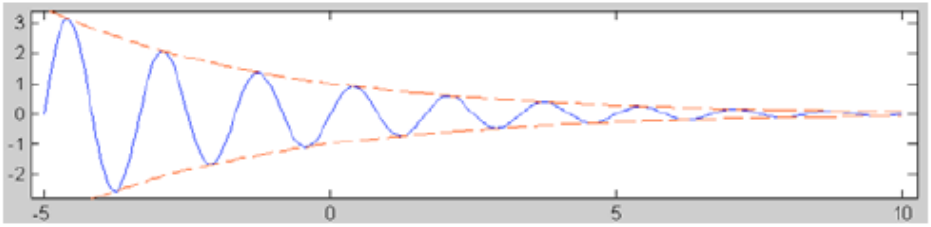
b) $x(t) = e^{-0.25t}$ (señal exponencial real).



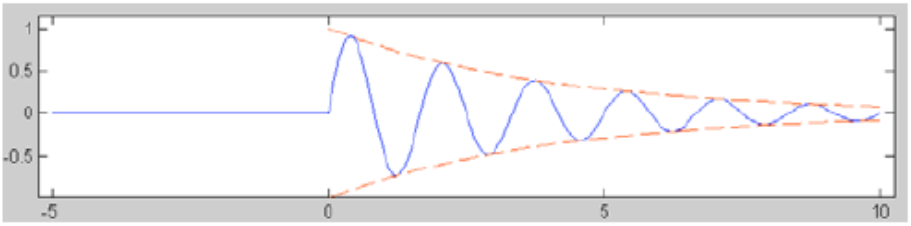
c) $x(t) = e^{-0.25t}u(t)$ (exponencial real que empieza en t=0).



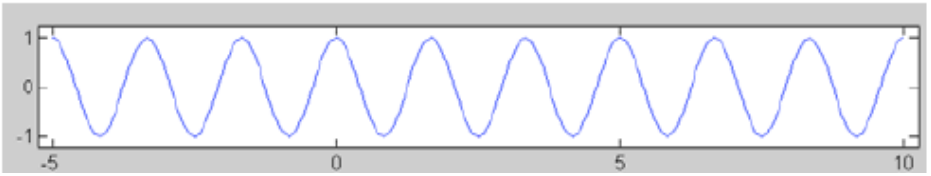
d) $x(t) = e^{-0.25t}\,\mathrm{sen}(6\frac{\pi}{5}t)$ (sinusoide amortiguada).

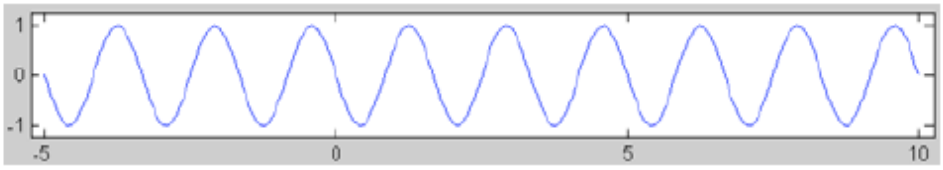


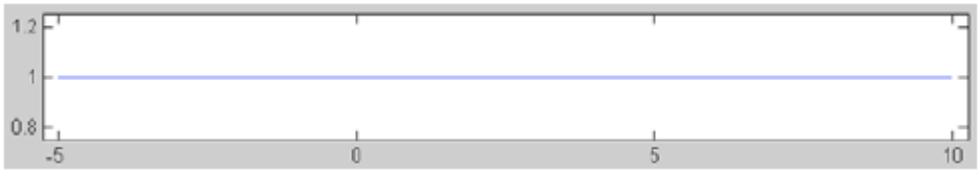
e) $x(t) = e^{-0.25t}\,\mathrm{sen}(6\frac{\pi}{5}t)u(t)$ (sinusoide amortiguada que empieza en t=0).

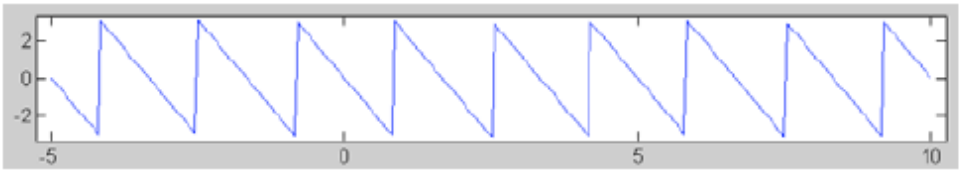


f) $x(t) = e^{-j6\frac{\pi}{5}t}$ (exponencial compleja).









Operaciones Simples con Señales

Ahora vamos a hacer algunas operaciones simples con una señal definida por su gráfica. Vamos a comenzar introduciendo en matlab la señal **x(t)** dada por este dibujo:

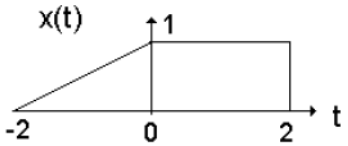


Figura 16.- Señal definida por una gráfica.

En principio, nos vale cualquier base de tiempos que abarque desde el –2 hasta el 2. Vamos a darle un poco de margen por ambos lados y la hacemos de –3 a 3 con incrementos de 0.05.

```
t = -3:0.05:3;
```

Hay que calcular la ecuación para la recta que va entre t = -2 y t = 0. Esta recta tendrá la ecuación típica: **r(t) = mt+b** donde:

- **m** es la pendiente. Al tratarse de una recta creciente debe resultar positiva. El valor de m se puede calcular como la tangente del ángulo que forma la recta con el eje horizontal (eje t). En este caso: $m = \frac{y1-y0}{x1-x0} = \frac{1-0}{0-(-2)} = 1/2$.
- **b** es el término independiente. Se puede calcular conociendo m y cualquier punto de la recta, por ejemplo **r(-2) = 0** ó **r(0) = 1** (con este último obtenemos una ecuación extremadamente simple que resulta **b = 1**). De hecho, el término independiente de una recta siempre es igual al punto en que corta al eje vertical (cuidado con esta propiedad porque, a veces, no es posible ver el punto de forma tan trivial como en este caso).

Por tanto, **x(t)** entre –2 y 0 (y sólo entre –2 y 0) es igual a la expresión: **t/2+1**. Por supuesto, **x(t)** es igual a 1 entre 0 y 2 e igual a cero en el resto de intervalos. Veamos como crear el vector de señal.

```
L = length(t); % Averiguar la longitud
x = zeros(1,L); % Primero todo ceros
p1 = find(t==-2); % Busco el -2
p2 = find(t==0); % Busco el 0
p3 = find(t==2); % Busco el 2
% Con el incremento elegido; -2, 0 y 2 estaran
% en la base de tiempos
x(p1:p2) = t(p1:p2)/2+1; % Parte de recta creciente
x(p2:p3) = 1; % Parte constante
% x en t=0 lo hemos calculado con dos formulas
% dando el mismo resultado
```

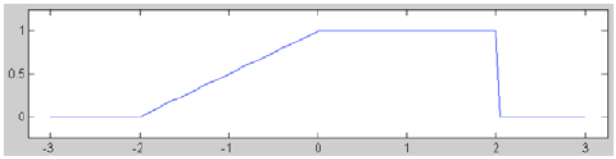


Figura 17.- Resultado de introducir x(t) en matlab.

Una vez que tenemos la señal x(t) en matlab vamos a realizar algunas operaciones simples con ella.

a) Traslación: calcular y representar $\mathbf{x_1(t) = x(t-3)}$.

Solución 1:

Sabemos que la señal es la misma pero todos los puntos se trasladrán de tiempo **t** a tiempo **t+3**. Esto es: basta con sumar 3 a la base de tiempos.

```
t1 = t + 3;
x1 = x;
```

Solución 2:

Movemos el vector de valores 3 unidades de tiempo hacia delante (y hacemos crecer la base de tiempos).

```
t_aux = 0.05:0.05:3; % Tres unidades de tiempo
% No empieza en cero porque lo vamos a
% añadir por la derecha
L3 = length(t_aux);
% Averiguar cuantos valores son los 3 segundos
x_aux = zeros(1,L3);
% Ceros para poner por la izda
x1 = [x_aux x]; % Los valores son los mismos
                % (con ceros por delante)
t1 = [t max(t)+t_aux];
% Crear nueva base de tiempos
```

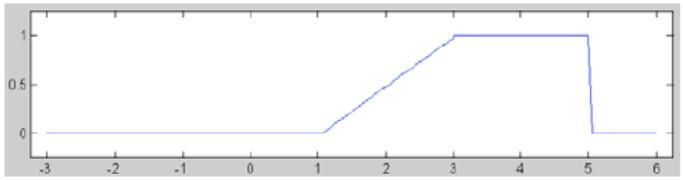


Figura 18.- Señal desplazada 3 segundos.

b) Escalado: calcular y representar $\mathbf{x_2(t) = x(2t)}$.

Sabemos que la señal es la misma pero "comprimida" (a la mitad). Por tanto, el punto situado en **t** pasará a **t/2**. Esto es: basta con dividir por 2 la base de tiempos.

```
t2 = t/2; % Tres unidades de tiempo
x2 = x;
```

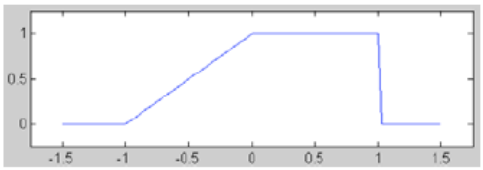


Figura 19.- Señal comprimida a la mitad.

Nótese que haciendo esto hemos dividido por dos el "incremento temporal" de la base de tiempos (los tiempos ahora están separados 0.025 segundos).

c) Reflexión: calcular y representar $\mathbf{x_3(t) = x(-t)}$.

Ahora se trata de que el punto situado en **t** pasará a **-t**. La primera idea es cambiar de signo la base de tiempos. Eso es correcto pero no es suficiente porque tanto la base de tiempos como el vector de valores están en orden inverso al que debieran. Eso debemos resolverlo invirtiendo ambos vectores.

```
t3 = -t; % Base de tiempos invertida
x3 = x; % Los valores son los mismos
final = length(t3);
t3 = t3(final:-1:1);
x3 = x3(final:-1:1); % Invertir
```

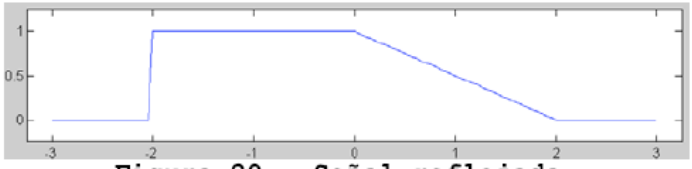


Figura 20.- Señal reflejada.

d) Sumar: sumar **x(t)** y $\mathbf{x_1(t)}$ (la señal del apartado a).

Para sumar dos señales no hay más que sumar los valores pero **DEBEMOS TENER LA MISMA BASE DE TIEMPO PARA AMBAS SEÑALES**. Si conservamos las variables **x1** y **t1** tenemos mucho hecho (se refiere a las variables del apartado a, solución 2). Viendo que **t1** es la misma base de tiempos de **x** pero extendida 3 undidades de tiempo no hace falta más que extender igualmente la base de tiempos de **x** (y añadir el número adecuado de ceros a los valores de **x**).

```
Lextra = length(t1)-length(t);
% Diferencia de longitudes temporales
% (en numero de valores)
t = t1; % Extendemos la base de tiempos de x
x = [x zeros(1,Lextra)]; % Añadimos ceros a x
t4 = t;
x4 = x+x1; % Calcular la señal suma (x4)
```

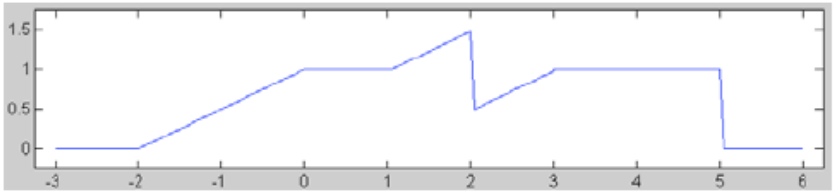
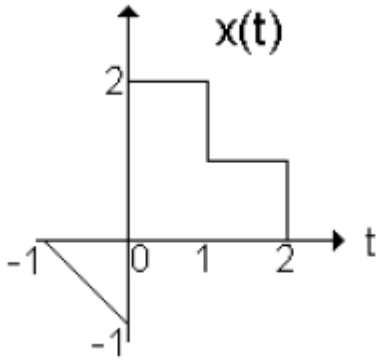


Figura 21.- Señal suma.

Generar x(t) de la figura y calcular y representar las señales x1(t)=(t-2), x2(t)=x(t/3), x3(t)=x(-t), x4(t)=x(t)-x1(t)



Concepto de Sistema

Ahora vamos a realizar en matlab un par de sistemas sencillos. Los textos de teoría definen un sistema como cualquier ente capaz de transformar señales. Para afianzar la idea de que un sistema transforma señales vamos a usar el concepto de "función de matlab" (que ya conocemos de la práctica introductoria) para crear nuestros primeros sistemas.

Un sistema lo vamos a definir como una función que podemos crear con el editor de matlab (o con cualquier otro editor) utilizando una cabecera del tipo:

```
function [y,ty] = Nombre(x,tx)
%
% Instrucciones que generen "y" y "ty"
% (a partir de x y tx).
%
% Estas instrucciones se graban en "Nombre.m"
%
```

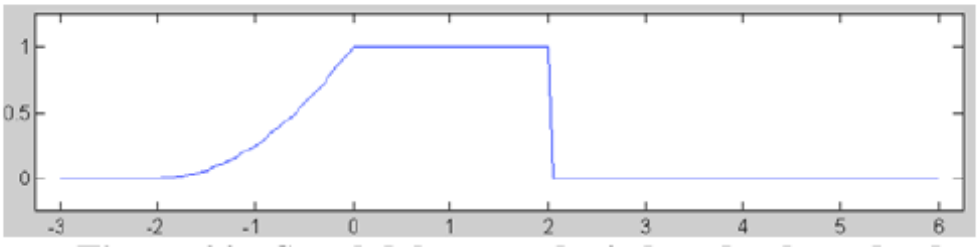
Nótese que hemos querido reforzar la idea de que una señal en matlab es una base de tiempos y un vector de valores. La función recibe como argumentos los dos vectores de la señal **x(t)** (entrada) y devuelve como resultados los dos vectores de la señal **y(t)** (salida).

Por ejemplo, para implementar el sistema que eleva al cuadrado ($\mathbf{y(t)=[x(t)]^2}$) podemos usar la función:

```
function [y,ty] = Cuadrado(x,tx)
ty = tx; % La base de tiempos es la misma
y = x.^2; % Los valores se elevan al cuadrado
```

Pruebe esta función (mantenga los valores de t y x

```
[y, ty]=cuadrado(x, t);
```



Señal elevada al cuadrado

DISEÑO DE UNA INTERFAZ GRAFICA USANDO *GUIDE* PARA GENERAR UNA *GUI* (GRAPHIC USER INTERFACE)

Preparación

Antes de asistir al laboratorio Ud. Debe:
-Ver el demo que tiene Matlab sobre Diseño de GUIs
-Leer los help que posee Matlab sobre el mismo tema
-Leer toda la información que se ofrece en este archivo
-Tener a disposición (en un disco) el logo de la UABC y la rutina fftplot. Esta rutina permite determinar el contenido espectral de una señal y graficar o su espectro o su Densidad Espectral de Potencia. Asegúrese de comprenderla.

Conceptos sobre Diseño de Interfaces gráficas (GUI)

Introducción:

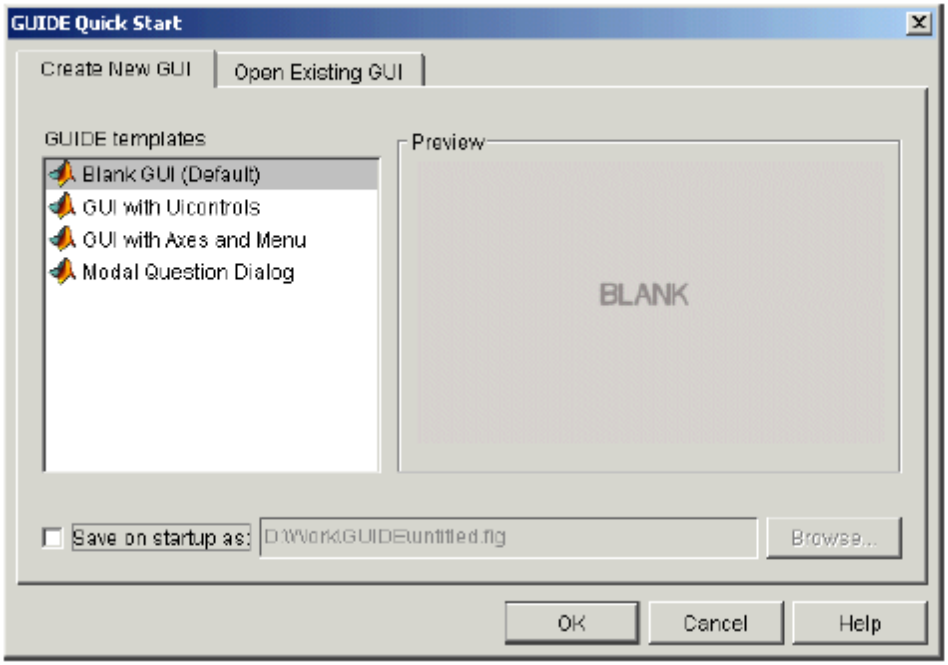
GUIDE es un ambiente de desarrollo que permite crear interfaces gráficas con el usuario, tales como botones y ventanas de selección, ventanas gráficas, menús, ejes para graficar, etc. Cuando en el
´command window´ se escribe guide, se ofrece la posibilidad de abrir hojas de trabajo ya creadas (p.ej.
>>guide archivo.fig) o una nueva sobre la cual se irán agregando componentes. Lo que se cree aquí se
guardará con la extensión **.fig.**
La primera vez que uno salva la interfaz que está diseñando se crea también un archivo **.m** sobre el cual habrá que programar lo que se quiere ver o controlar desde el GUI
Una vez que se diseña la interfaz gráfica (GUI) que uno desea fijando las características de botones,
ventanas, etc. que la conforman, se puede entonces programar dicha interfaz con el editor de archivos
.m

Herramientas:

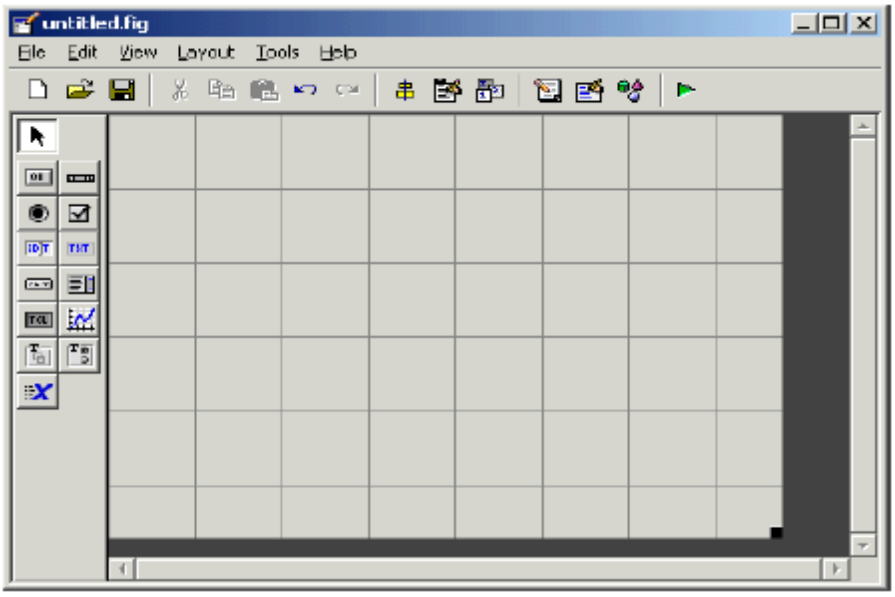
Uno puede seleccionar botones, menús desplegables, graficas, etc. y transferirlos a la hoja de trabajo en una posición determinada. Haciendo doble clic sobre algún elemento y luego, con el cursor
cambiado a una X, uno se posiciona en la hoja de trabajo y selecciona el tamaño que quiere que ocupe
la herramienta seleccionada dentro de la hoja de trabajo.
Cuando uno selecciona una herramienta, con doble clic se pueden seleccionar todos los detalles: Por
ejemplo el color de fondo, los nombres con los cuales se les relacionará en el archivo .m, etc.

A diferencia de la ejecución de funciones o scripts de MATLAB tradicionales, la ejecución de GUIs viene dada por la secuencia de eventos y las acciones asociadas a cada uno de ellos. El script no
finaliza a menos que el usuario genere un botón que ejecute tal acción. El desarrollo de GUIs se realiza en dos etapas:

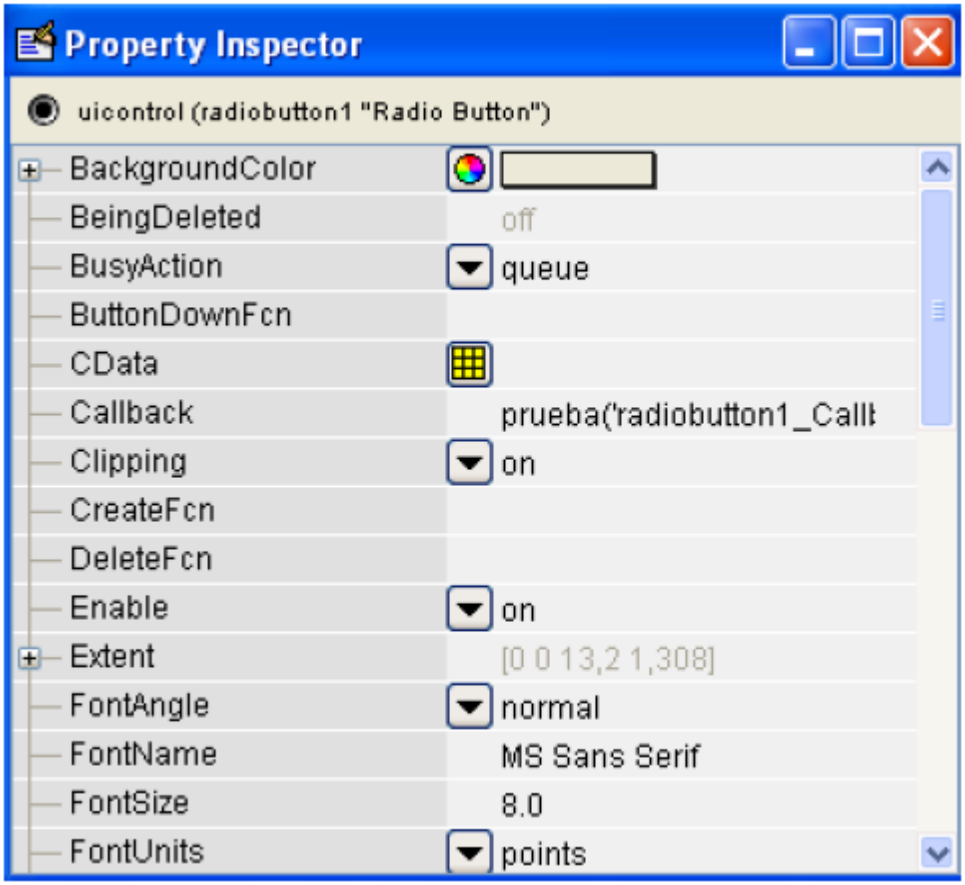
a) Selección de las herramientas (controles, menús y ejes) que conformarán el GUI.
b) Codificación de la respuesta de cada uno de los componentes ante la interacción del usuario.
Lo primero que se debe hacer es escribir *guide* en el *command window*. Aparecerá la siguiente ventana. Se selecciona una hoja en blanco (Blank GUI)



Aparecerá de inmediato una hoja de trabajo como la que sigue:



A la izquierda están las herramientas disponibles. Si en este momento se salva el GUI debemos seleccionar un nombre. Se almacenará en el directorio donde estemos ubicados **nombre.fig** y **nombre.m** .
Para utilizar una herramienta se selecciona y arrastra a la posición deseada en la hoja de trabajo; se puede modificar su tamaño tal y como se hace con cualquier figura. Otras características se pueden
modificar haciendo doble click. Aparecerá un editor de propiedades como el que sigue:
Al hacer una modificación en el editor de propiedades también cambiará el código relacionado a cada botón en el archivo **.m**



El archivo **nombre.m** tiene toda una estructura de handles (manejadores) que alimentarán a la GUI. La estructura de handles es pasada como una entrada a cada callback (llamada a una parte de un
programa).
Puedes usar la estructura de handles para:
Compartir datos entre callbacks
Acceder la data en el GUI

Para almacenar los datos contenidos en una variable X, se fija un campo de la estructura de handles
igual a X y se salva la estructura de handles con guidata como se muestra a continuación:
handles.current_data=X
guidata(hObject, handles)
En cualquier momento puedes recuperar la data haciendo
X=handles.current_data

Código asociado a cada elemento del GUI

Automáticamente al crear el GUI y salvarlo aparece en el archivo .m una cantidad de líneas de código
fijas. En la primera parte del script aparece una cantidad de líneas de código fijo. La primera instrucción :
function varargout = untitled(varargin) indica que se está creando un GUI de nombre untitled con argumentos de salida varargout y argumentos de entrada varagin. Solo se muestran las dos primeras y
la última línea.

```
function varargout = untitled(varargin)
% UNTITLED M-file for untitled.fig
……..
% Last Modified by GUIDE v2.5 19-Oct-2005 16:25:55
```

Aquí comienza un código de inicialización que se pide no se edite. También se muestra solo la primera
y última línea

```
% Begin initialization code - DO NOT EDIT
```

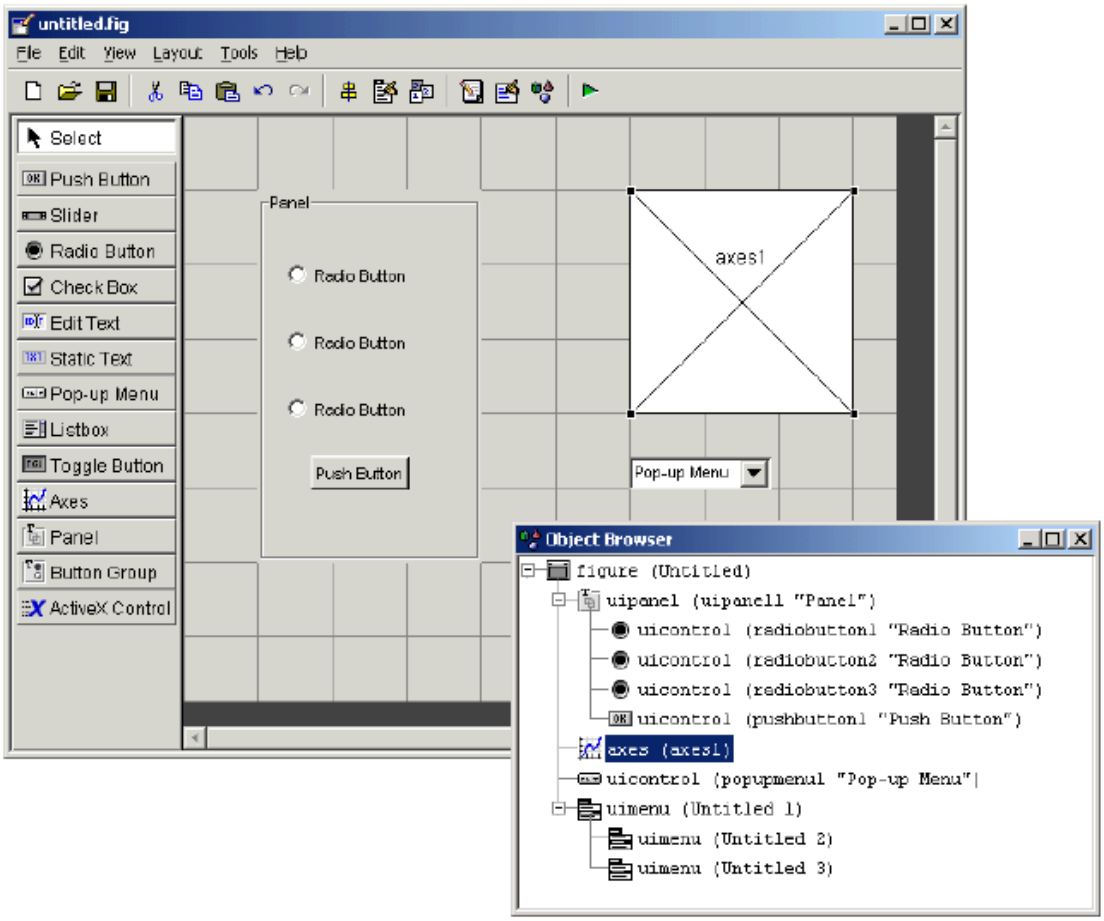
4

```
……
% End initialization code - DO NOT EDIT
```

Hasta aquí llega el código de inicialización. Comienza entonces lo que queremos que ocurra antes de que el GUI se haga visible. Estas instrucciones se colocan después de la línea donde dice
function untitled_.....

```
% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
…..
……
varargout{1} = handles.output;
```

Luego de esto aparecen los callback’s dependiendo de las herramientas que se han incorporado al GUI.
Por ejemplo:



Aquí se han colocado 3 Radio Button, 1 Push Button, un Pop-Up menú y 1 eje para graficar.
A continuación describiremos brevemente las herramientas disponibles, el código automático asociado
a cada una de ellas y como interactuar con las mismas.

boton pushbutton: Se ejecuta una determinada acción cuando son presionados. En el archivo **.m**
aparecen automáticamente un grupo de instrucciones asociadas a él.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
5
AQUI SE COLOCA EL CODIGO QUE SE ESPERA EJECUTAR CUANDO SE PRESIONES
EL
PUSHBUTTON1
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

listas de seleccion: Aquí se puede colocar una lista de elementos para que el usuario pueda seleccionar alguno. En el archivo **.m** aparecen automáticamente un grupo de instrucciones asociadas a
él.

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
……
% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
……..
% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
% contents{get(hObject,'Value')} returns selected item from listbox1
Como se observa en la ayuda (Hint) que aparece en las dos últimas líneas, si se coloca la instrucción
Contents= get(hObject,'String')
Se obtendrá un número que indica que selección se hizo. Por ejemplo si se seleccionó el tercer elemento de la lista, Contents dará 3

Boton edit

Permite a los usuarios ingresar o modificar parámetros que se quieren introducir.
function edit1_Callback(hObject, eventdata, handles)
…..
% Hints: get(hObject,'String') returns contents of edit1 as text
% str2double(get(hObject,'String')) returns contents of edit1 as a double
Observe la ayuda (HINT)
SI SE COLOCA LA INSTRUCCION
A= str2double(get(hObject,'String'))
Entonces, se podrá tener en A el valor del número que uno escribió en la casilla
SI SE COLOCA LA INSTRUCCION
A= get(hObject,'String')
Entonces, se podrá tener en A los caracteres que uno escribió

Botón RadioButton

Son botones de selección. Si hay varios generalmente son mutuamente excluyentes. Para seleccionarlo
basta ubicarse y presionar el ratón.
% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% Hint: get(hObject,'Value') returns toggle state of radiobutton1
SI SE COLOCA LA INSTRUCCION
a=get(hObject,'Value')
Entonces en a se tendrá 1 si el botón fue seleccionado

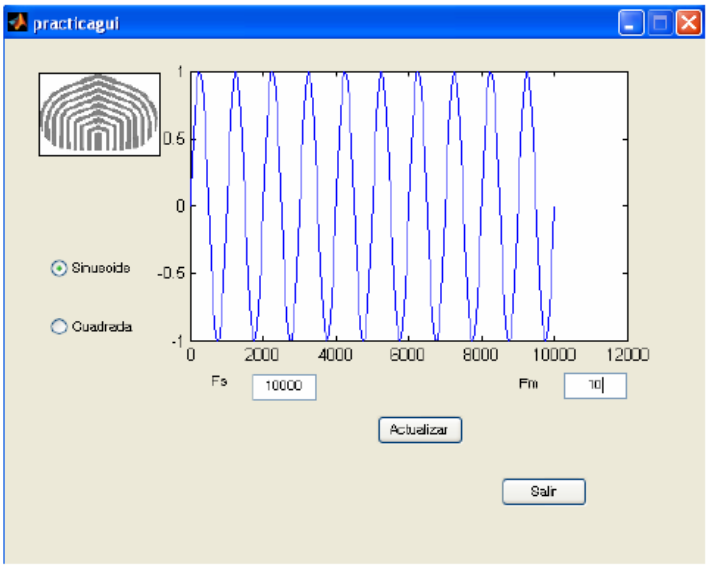
Ejes para graficar:

Aquí no se genera nada especial pero uno debe fijar las condiciones de la gráfica y activarla o desactivarla según convenga.
Por ejemplo si la gráfica tiene asociado el nombre de axes1 y se quiere deshabilitar
set(handles.axes1,'Visible','off');

PopUp menú. Cuando se hace click despliega opciones. Para agregar elementos a la lista, en el
editor de propiedades se busca el elemento string y allí se coloca la lista de las opciones.
% --- Executes during object creation, after setting all properties.

function popupmenu1_CreateFcn(hObject, eventdata, handles)
…….
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
……..
% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
% contents{get(hObject,'Value')} returns selected item from popupmenu1
SI SE COLOCA LA INSTRUCCION
contents=get(hObject,'Value')
ENTONCES SE OBTENDRA UN NUMERO QUE INDICA QUE SELECCIÓN SE HIZO.
POR
EJEMPLO SI SE SELECCIONÓ EL TERCER ELEMENTO DE LA LISTA, contents DARA 3
7

Experimento



Ud. Debe diseñar una GUI con las siguientes características:
• Debe presentar del lado izquierdo el sello y nombre de la UABC
• Permitirá dibujar 1 de dos señales (una sinusoide y una cuadrada)
• La selección del tipo de señal se hará con radiobutton.
• El usuario debe fijar el valor de Fs y Fseñal y utilizar un botón para actualizarlos
• En cualquier momento se puede cambiar el tipo de señal.
• Debe tener un botón para salir del programa que cierra todas las ventanas
Para esto desde el comand window escriba *guide*. Debe escoger Blank Gui o sea una hoja de trabajo
en blanco con las herramientas del lado izquierdo. Seleccione el icono de gráficas y arrástrelo hacia la
hoja de trabajo. Este se llamará axes1. Arrastre otro; este se llamará axes2. En estos lugares Ud. colocará el sello de la UABC (axes2) y la gráfica (sinusoide o cuadrada) (axes1); dándole doble click

sobre esta puede ver sus propiedades; busque la variable Tag y cambie el nombre de axes2 a Imagen.
Guarde el archivo con el nombre practicaGui. Si Ud. quiere deshabilitar inicialmente axes 1 debe escribir (luego del grupo de instrucciones que se ejecutan antes de que el GUI sea visible)

```
set(handles.axes1,'Visible','off');
```

Esto “apaga” la rejilla donde se graficará la señal seleccionada
La instrucción SET tiene la siguiente estructura:
SET(H,'PropertyName',PropertyValue)
Luego se habilitará Imagen que es donde aparecerá el sello de la UABC.

```
axes(handles.Imagen);
image(imread('usb','jpg'));
```


Además eliminará las escalas en X y en Y

```
set(handles.Imagen,'Xtick',[]);
set(handles.Imagen,'Ytick',[]);
```

Ahora arrastre un radiobutton que se llamará radiobutton1. Actívelo agregando, en la zona de código que se ejecuta antes de que la interfaz sea visible, la siguiente instrucción:

```
set(handles.radiobutton1,'Value',1);
```

Ahora arrastre otro radiobutton que se llamará radiobutton2. Inicialmente desactívelo:

```
set(handles.radiobutton2,'Value',0);
```

En este momento aparecerán dos callback cada uno asociado a cada radiobutton. En el callback de radiobutton1 agregue las siguientes instrucciones:

```
set(handles.radiobutton1,'Value',1);
set(handles.radiobutton2,'Value',0);
```

En el callback de radiobutton2 coloque, en cambio:

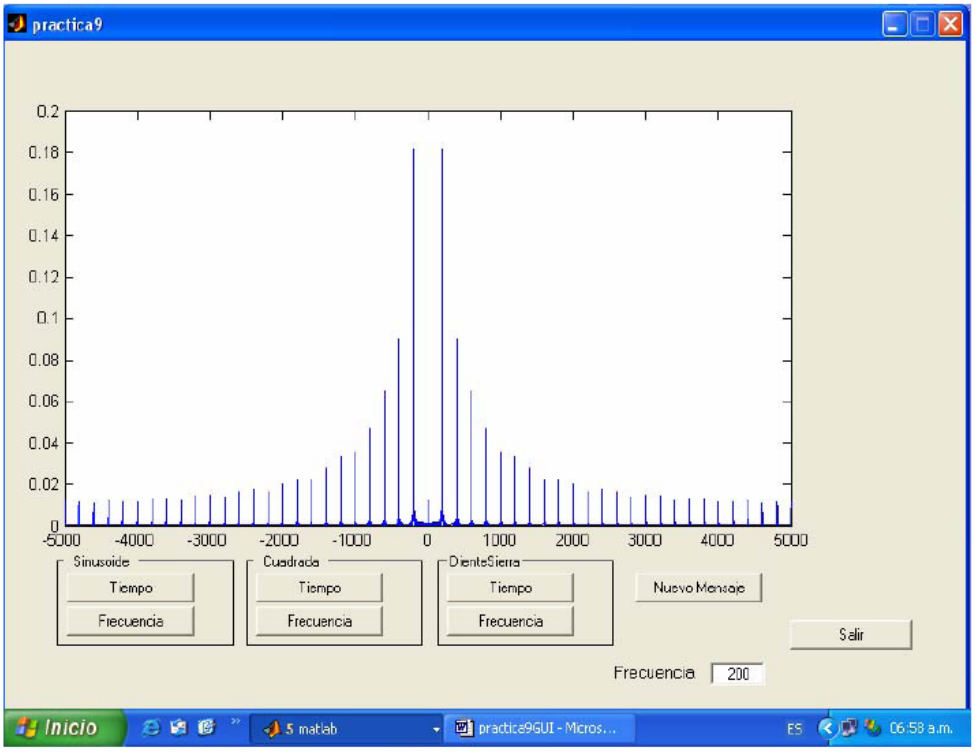
```
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton2,'Value',1);
```

Esto los hará excluyentes.
Si en este momento ejecuta el archivo **.m** le aparecerán: el sello de la UABC y los dos radiobutton
estando activado el 1 (Sinusoide). Ahora le pondrá nombres al lado de los radio button. Para eso hará
doble clic sobre el radio button y aparecerá el Inspector de Propiedades (Property Inspector); busque
String y allí pondrá el nombre deseado (**Sinusoide**). Haga lo mismo con el otro y póngale **Cuadrada.**
Agregue ahora un pushbutton: Su tag será pushbutton1 y al String pongale **“Salir”.** En donde está el código correpondiente al pushbutton escriba la instrucción “close all”. Cuando se presione el
botón Salir se cerrarán todas las ventanas incluyendo el **.fig** que se está ejecutando

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

Ahora generará el vector de tiempo y los dos mensajes de la siguiente manera (coloque fs y fm a

su gusto, por ejemplo10000 y 10). Este código puede colocarlo inmediatamente después de donde el
comentario dice: % --- Executes just before NOMBRE DE SU PROGRAMA is made visible.
9
```
t=[0:1/fs:1];
mensaje1=sin(2*pi*fm*t);
mensaje2 = square(2*pi*fm*t);
handles.mensaje1 = mensaje1;
handles.mensaje2 = mensaje2;
Se habilita el sitio donde se graficará la función
set(handles.axes2,'Visible', 'on');
axes(handles.axes2)
```
AQUI SE SELECCIONA SI SE TRABAJARA CON LA SINUSOIDE O CON LA CUADRADA
DE ACUERDO AL BOTON QUE ESTE SELECCIONADO Y SE GRAFICA
```
switch get(handles.radiobutton1,'Value')
case 1
mensaje = handles.mensaje1;
case 0
mensaje = handles.mensaje2;
end
plot(mensaje)
```
Ahora ejecute y vea lo que ocurre. Arregle el código para que la sinusoide aparezca desde el comienzo.
Añada ahora la posibilidad de agregarle a la GUI dos ventanas que permitan fijar la frecuencia de muestreo y la frecuencia de las sinusoides y la cuadrada.
10
Ejercicio



• Ud. Debe diseñar una GUI similar a la siguiente:
• Esta interfaz le permitirá presentar 3 señales distintas o en tiempo o en frecuencia. La frecuencia(fm) de la Sinusoide, la fundamental de la Cuadrada y la fundamental de la Diente de Sierra podrá ser fijada por el usuario; cuando se cambie la frecuencia se presionará el botón Nuevo Mensaje para que esta sea incorporada como dato; luego se puede visualizar con la frecuencia cambiada. Se debe disponer de un botón de Salir que cierre todas las ventanas.
• Antes de que aparezca la primera señal debe aparecer el sello de la UABC al lado izquierdo. Cuando se grafique la primera señal este desaparecerá.
• Luego de que presione cualquiera de los botones que indican Tiempo o Frecuencia aparecerá la primera señal o el primer espectro graficado.
• En cualquier momento Ud. podrá cambiar fm, Presionar nuevo mensaje y estará listo para poder
ver otra Sinusoide, Cuadrada o Diente de Sierra y sus espectros.
• Una vez que tenga operativa esta GUI, debe agregarle nuevas ventanas que fijen la frecuencia de muestreo y el número de puntos que Ud. desea trabajar. (fs y N).
• Posteriormente piense en un diseño diferente en el cual las señales (sinusoidal, cuadrada y Diente de sierra) se puedan seleccionar por medio de un menú del tipo listbox o pop-up menú, mientras que el dominio de presentación sea elegido con un radio button.

SEÑALES Y SISTEMAS

Práctica 4.

Solución de ecuaciones de diferencia por recursión

1. Simule en el Matlab la ecuación

$$y[n] - \left(1 + \frac{I}{12}\right) y[n-1] = -x[n], \qquad n = 0, 1, 2, \ldots \ldots$$

Escriba el programa M que resuelve la siguiente ecuación.

Considere I= 0.12, x=200 y y(0)=6000

```
%ecuacion de diferencia (ejercicio)
% Calcula y(n)
y0=input ('y ');
I=input ('constante  ');
x=input (' entrada constante ');
y=[];  %se define a y como un vector vacio
y(1)=(1+(I/12))*y0-x;
for n=2:360
   y(n)=(1 + (I/12))*y(n-1)-x;
   if  y(n) < 0,  break, end
end
```

*Actividad a realizar posterior a la práctica: Modifique el problema de forma que pueda tener una entrada arbitraria x diferente de una constante

2. Haga un programa en el Matlab que resuelva la ecuación de diferencias:

$$m[n] = e(n) - e(n-1) - m(n-1)$$

dada la entrada e(n) dada por:

$$e[n] = \begin{cases} 1 & si\ es\ n\ \ par \\ 0 & si\ impar \end{cases}$$

Grafique la respuesta y la entrada. Realice los misma simulación utilizando el Simulink

Practica # 5 Convolución Discreta y respuesta la impulso

NOTAS PRÁCTICAS SOBRE MATLAB

En cualquier momento, puede obtener ayuda sobre una función Matlab introduciendo en la consola el comando help <funcion>.

Dependiendo del resultado de cada apartado, puede que se le solicite representarlo de forma gráfica, para lo cual pueden serle de ayuda los comandos siguientes:

- plot o stem para representar gráficamente un conjunto de valores.
- subplot para representar conjuntamente más de una gráfica en la misma ventana.
- figure para crear una nueva ventana gráfica y no sobrescribir la gráfica de la ventana anterior.
- title, xlabel, ylabel para insertar texto en el encabezado, en el eje horizontal y en el eje vertical, respectivamente, de la gráfica activa.

1. Introducción a la convolución

La función de Matlab conv calcula la suma de convolución

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

Para calcular la suma, Matlab requiere que $x[n]$ y $h[n]$ sean secuencias de duración finita. Si asumimos que $x[n]$ es no nula solamente en el intervalo $n_x \le n \le n_x + N_x - 1$ (siendo N_x su longitud) y que $h[n]$ es no nula solamente en el intervalo $n_h \le n \le n_h + N_h - 1$ (siendo N_h su longitud), entonces $y[n]$ es no nula únicamente en el intervalo $(n_x + n_h) \le n \le (n_x + n_h) + N_x + N_h - 2$ (siendo su longitud $N_x + N_h - 1$). Esto significa que conv solamente necesita calcular $y[n]$ para las $N_x + N_h - 1$ muestras de este intervalo.

Si x es un vector N_x-dimensional que contiene las muestras de $x[n]$ en el intervalo $n_x \le n \le n_x + N_x - 1$ y h es un vector N_h-dimensional que contiene las muestras de $h[n]$ en

el intervalo $n_h \le n \le n_h + N_h - 1$, entonces $y = conv(x,h)$ devuelve en y las $N_x + N_h - 1$ muestras de $y[n]$ en el intervalo $(n_x + n_h) \le n \le (n_x + n_h) + N_x + N_h - 2$.

Ha de tener en cuenta que la función conv no devuelve los índices de las muestras de $y[n]$ almacenadas en el vector y. El usuario de la función conv es el responsable de conocer cuáles son dichos índices en función de los índices de los vectores de entrada.

EJEMPLO

Convolución de pulsos rectangulares.

Suponga que *x[n]* y *v[n]* son iguales al pulso rectangular definido por:

$$p[n] = \begin{cases} 1, & 0 \le n \le 9 \\ 0, & otro\ valor\ de\ n \end{cases}$$

Encuentre el producto de convolución utilizando la función de matlab ***conv***.

Solución:
Los comandos en MATLAB para calcular la convolución para este caso son:

```
p=[0 ones(1,10) zeros(1,5)]; % corresponde para n=-1 hasta n=14
  x=p; v=p;
  y= conv(x,v);
  n=-2:25;
  stem(n,y(1:length(n)))
```

a)Dibuje con la instrucción **stem** la gráfica y etiquetela con title, xlabel, ylabel, es decir póngale una nombre a los ejes y título a la gráfica.

b) Determine N (Nx)_____, M (Nh)_____ y el largo del dominio (no cero) de la señal resultante de la convolución

c) Determine la convolución de la señal resultante consigo misma y a la vez haga lo mismo con la salida

d) A lo largo del ejercicio emplearemos las tres señales siguientes:

$$x_1[n] = \begin{cases} 1 & n = 0 \\ 2 & 1 \le n \le 4 \\ 0 & resto \end{cases}$$

$$h_1[n] = \begin{cases} -1 & n = 0 \\ 1 & n = 1 \\ 2 & n = 2 \\ 0 & resto \end{cases}$$

$$h_2[n] = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ 3 & n = 3 \\ -1 & n = 4 \\ 0 & resto \end{cases}$$

Realice

a) x1*h1+x1*h2, x1*(h1+h2)

b) X1*(h1*h2) , (x1*h1)*h2, (x1*h2)*x1

Grafique los resultados con su valor adecuado de “n” y escriba sus conclusiones

Práctica 6

Series de Fourier

Definiciones. Sea $f : \mathbb{R} \to \mathbb{R}$ una función continua a trozos y periódica de período T y sea $\omega = 2\pi/T$. Los *coeficientes de Fourier* de f son los números definidos por

$$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega t)\,dt \qquad \text{para } n = 0, 1, 2\ldots,$$

y

$$b_n = \frac{2}{T}\int_0^T f(t)\text{sen}\,(n\omega t)\,dt \qquad \text{para } n = 1, 2, \ldots$$

Puesto que la función tiene período T, las integrales anteriores pueden hacerse sobre cualquier intervalo de longitud T, por ejemplo $[-T/2, T/2]$.

La serie $\frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left[a_n\cos(n\omega t) + b_n sen\,(n\omega t)\right]$, donde los a_n y los b_n son los correspondientes coeficientes de Fourier de una función f, se llama *desarrollo en serie de Fourier*, o simplemente *serie de Fourier*, de f. Para indicar que una serie trigonométrica es la serie de Fourier de una función dada se suele escribir

$$f(t) \sim \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left[a_n\cos(n\omega t) + b_n\text{sen}\,(n\omega t)\right].$$

Condiciones de Dirichlet. Sea $f : \mathbb{R} \to \mathbb{R}$ una función periódica de período T. Se dice que f satisface las *condiciones de Dirichlet* si en cada período la función $f : [0, T] \to \mathbb{R}$ es continua salvo en un número finito de discontinuidades todas ellas de salto y sólo tiene una cantidad finita de máximos y mínimos locales estrictos. Puede probarse, en particular, que si una función periódica es tal que ella y su derivada están definidas y son continuas salvo un número finito de discontinuidades de salto, entonces dicha función verifica las condiciones de Dirichlet. Prácticamente todas las funciones —señales— de interés en las aplicaciones las verifican.

Teorema de convergencia de Dirichlet. Sea $f : \mathbb{R} \to \mathbb{R}$ una función periódica de período T que satisface las condiciones de Dirichlet y sea

$$f(t) \sim \frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(n\omega t) + b_n \text{sen}\,(n\omega t)\right],$$

con $\omega = 2\pi/T$, su serie de Fourier.

(1) Si f es continua en un punto t, entonces la serie de Fourier converge en ese punto a $f(t)$, o sea,

$$\frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(n\omega t) + b_n \text{sen}\,(n\omega t)\right] = f(t).$$

(2) Si f tiene una discontinuad de salto en un punto t, entonces la serie de Fourier converge en ese punto al punto medio del salto, o sea,

$$\frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(n\omega t) + b_n \text{sen}\,(n\omega t)\right] = \frac{f(t^-) + f(t^+)}{2},$$

donde, como es habitual, $f(t^-) = \lim_{\tau \to 0, \tau > 0} f(t-\tau)$ indica el límite de f en t por la izquierda y $f(t^+) = \lim_{\tau \to 0, \tau > 0} f(t+\tau)$ indica el límite de f en t por la derecha.

El teorema nos dice, en particular, que si f satisface las condiciones de Dirichlet y redefinimos el valor de f en cada punto de discontinuidad como el punto medio del salto, o sea, poniendo $f(t) = \dfrac{f(t^-) + f(t^+)}{2}$, entonces la suma de la serie de Fourier coincide con $f(t)$ en cada $t \in \mathbb{R}$. Por eso, en lo que sigue, y salvo que se diga lo contrario, supondremos que esto se cumple.

Desarrollos de medio intervalo. Sea $f : [0, L] \to \mathbb{R}$ una función continua a trozos. Sea $T = 2L$, entonces la *extensión T-periódica y par de f* es la función definida en el intervalo $[-L, L] = [-T/2, T/2]$ por

$$f_{\text{par}}(t) = \begin{cases} f(-t) & \text{si } -L \le t \le 0, \\ f(t) & \text{si } 0 \le t \le L, \end{cases}$$

y extendida periódicamente a toda la recta real $\mathbb{R}$. Obviamente, f_{par} es una función par, así que su desarrollo en serie de Fourier es una serie de cosenos que se conoce, en este contexto, como el *desarrollo de f en serie de cosenos en medio intervalo* $[0, L]$ y viene dado por

$$f(t) \sim \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(n\omega t) \qquad \text{en } [0, L],$$

siendo $\omega = \pi/L$ y

$$a_n = \frac{2}{L}\int_0^L f(t)\cos(n\omega t)\,dt \quad (n = 0, 1, 2, \dots).$$

La *extensión T-periódica e impar de f* es la función definida en $[-L, L] = [-T/2, T/2]$ por

$$f_{\text{impar}}(t) = \begin{cases} -f(-t) & \text{si } -L \leq t \leq 0, \\ f(t) & \text{si } 0 \leq t \leq L, \end{cases}$$

y extendida periódicamente a toda la recta real $\mathbb{R}$. Obviamente, f_{impar} es una función impar, así que su desarrollo en serie de Fourier es una serie de senos que se conoce, en este contexto, como el *desarrollo de f en serie de senos en medio intervalo* $[0, L]$ y viene dado por

$$f(t) \sim \sum_{n=1}^{\infty} b_n \operatorname{sen}(n\omega t) \qquad \text{en } [0, L],$$

siendo $\omega = \pi/L$ y

$$b_n = \frac{2}{L} \int_0^L f(t) \operatorname{sen}(n\omega t)\, dt \quad (n = 1, 2, \dots).$$

Ejercicio resuelto 1. Considera la función $f : [0, 2] \to \mathbb{R}$ dada por $f(x) = 1 - x/2$. Un cálculo elemental muestra que los coeficientes del desarrollo en serie de Fourier de senos de f vienen dados por $b_n = \dfrac{2}{n\pi}$. Diseña una función de Matlab que dibuje la suma parcial de los N primeros sumandos de la serie de Fourier de senos de la función y muestre simultáneamente la gráfica de f.

Solución. Basta crear la siguiente función en un archivo llamado `fousen.m`. Obsérvese la estructura de la línea que define la función f.

```
function fousen(N)
x=-2:0.005:2;
sumparcial=0;
b=zeros(1,N);
for k=1:N
b(k)=2/(k*pi);
sumparcial=sumparcial+b(k)*sin(k*x*pi/2);
end
f=(x<0).*(-1-x/2)+(x>=0).*(1-x/2);
plot(x,f,'b',x,sumparcial,'g'),shg
```

En el ejercicio anterior hemos obtenido explícitamente el valor de los coeficientes de Fourier de la función f. Usando Matlab podemos aproximar dichos coeficientes mediante las diferentes funciones de integración numérica entre las que destacamos quad y `quadl`.

Ejercicio resuelto 2. Considera la función 2-periódica $f : \mathbb{R} \to \mathbb{R}$ tal que $f(x) = x^2$ si $-1 < x < 1$. Diseña una función de Matlab que aproxime numéricamente los N primeros coeficientes de la serie de Fourier de la función y que dibuje la suma parcial frente a f.

Solución. Puesto que la función es par, tenemos que

$$a_0 = \frac{2}{3}, \quad a_n = \frac{2}{2}\int_{-1}^{1} f(t)\cos(n\pi t)\,dt = 2\int_0^1 t^2\cos(n\pi t)\,dt \qquad \text{para } n = 1, 2\ldots,$$

y

$$b_n = \frac{2}{2}\int_{-1}^{1} f(t)\text{sen}\,(n\pi t)\,dt = 0 \qquad \text{para } n = 1, 2, \ldots$$

De esta forma, la solución del problema viene dada por la siguiente función que debe ser almacenada en un archivo llamado `fou.m`.

```
function fou(N)
x=-1:0.05:1;
a=zeros(1,N);
sumparcial=1/3;
for k=1:N
 a(k)=quadl(@fun,-1,1,1e-9,[],k);
 sumparcial=sumparcial+a(k)*cos(k*x*pi);
end
f=x.^2;
plot(x,f,'b',x,sumparcial,'g'),shg
function y=fun(t,n)
y=(t.^2).*cos(n*pi*t);
```

Nota: Obsérvese el funcionamiento de la función `quadl` y los parámetros de los que depende. Pide ayuda a Matlab para entender su estructura.

```
>> salto=A(2,2)-A(1,2)
```

Ejercicio 5. Ejecute la función obtenida en el ejercicio 3 con $N = 4, 8, 16, 32$ y verifique el fenómeno de Gibbs en $x = 0$. Estime gráficamente el valor del salto.

Ejercicio 6. Repita los ejercicios 1 y 4 cambiando el desarrollo en serie de Fourier de senos de f por el desarrollo en serie de Fourier de cosenos comentando las diferencias observadas.

Ejercicio 3. Considera la función 2-periódica $f : \mathbb{R} \to \mathbb{R}$ tal que $f(x) = 1$ si $0 < x < 1$ y $f(x) = -1$ si $-1 < x < 0$. Calcula los coeficientes del desarrollo en serie de Fourier de f. Diseña una función de Matlab que proporcione los N primeros coeficientes de la serie de Fourier de la función y que dibuje la suma parcial frente a f.

El fenómeno de Gibbs

El teorema de Dirichlet nos dice que, en los puntos de discontinuidad, la gráfica de la suma de la serie de Fourier pasa por el punto medio del salto. Si se dibujan las sumas parciales se ve que en las cercanías de los puntos de discontinuidad se reduce la velocidad de convergencia de la serie y que la gráfica de la suma parcial oscila alrededor de la gráfica de la función. Cuando se aumenta el número de términos, las oscilaciones se condensan a ambos lados del punto pero su amplitud no parece decrecer. Esto se conoce como el *fenómeno de Gibbs*,en honor de J.W. Gibbs (1839–1903) que lo analizó en 1899 probando que la amplitud de la oscilación a cada lado de la gráfica de la función tiende a ser $\frac{1}{2\pi}\int_0^{\pi} \frac{\operatorname{sen}(t)}{t}\, dt - 1 \approx 0.0895$ veces el tamaño del salto, o sea, aproximadamente el 9% del tamaño del salto.

Ejercicio resuelto 4. Ejecute la función obtenida en el ejercicio 1 con $N = 4, 8, 16, 32$ y verifique el fenómeno de Gibbs en $x = 0$. Estime gráficamente el valor del salto. Para tal estimación, consulte previamente la ayuda de Matlab sobre la función `ginput`.

Solución. Hacemos uso de la función `ginput`.

```
>> fousen(32),A=ginput(2)
```

Igualdad de Parseval

Igualdad de Parseval. Sea f una función periódica de período T que verifica las condiciones de Dirichlet y sea

$$f(t) \sim \frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(n\omega t) + b_n \text{sen}\,(n\omega t)\right]$$

su serie de Fourier. Entonces

$$\frac{1}{T}\int_0^T \left[f(t)\right]^2 \, dt = \frac{1}{4}a_0^2 + \frac{1}{2}\sum_{n=1}^{\infty}\left[a_n^2 + b_n^2\right]. \qquad (*)$$

En particular, la serie de los cuadrados de los coeficientes de Fourier es convergente.

La integral $P = \dfrac{1}{T}\displaystyle\int_0^T \left[f(t)\right]^2 \, dt$ se llama *media cuadrática* o *potencia media* de f. Por ejemplo, si $f(t)$ representa el voltaje que se aplica a una resistencia de 1 ohmio, entonces la potencia media de f coincide con la potencia eléctrica media (energía por unidad de tiempo medida en watios) disipada por la resistencia en cada período. Ahora bien, la potencia media de cada uno de los armónicos presentes en la señal es

$$P_0 = \frac{1}{T}\int_0^T \left[a_0/2\right]^2 \, dt = \frac{1}{4}a_0^2,$$
$$P_n = \frac{1}{T}\int_0^T \left[a_n \cos(n\omega t) + b_n \text{sen}\,(n\omega t)\right]^2 \, dt = \frac{1}{2}a_n^2 + \frac{1}{2}b_n^2 \quad (n = 1, 2, \dots),$$

luego la igualdad de Parseval nos dice que la potencia media de la señal es la suma de las potencias medias de sus componentes armónicos, $P = \displaystyle\sum_{n=0}^{\infty} P_n$.

Ejercicio 7. Diseñe una función de Matlab que aproxime la potencia media de la función del ejercicio 3. Dicha función debe depender del número de sumandos de la serie $(*)$.

Práctica 7

FILTRADO Y CONVOLUCION

Preparación

Antes de asistir al laboratorio Ud. Debe:

- Leer en el Help de Matlab las siguientes funciones:
- **fft, ifft,conv,filter,filtfilt,sosfilt**
- Leer sobre filtros FIR e IIR

Ejercicio Nro. 1

Sea la secuencia **x[n] = cos(0.25πn) + cos(0.5πn)+ cos(0.52πn)**. Se pide Calcular la DFT utilizando la función matlab fft(x,N) con N=L= longitud de las secuencia x[n] y representar su módulo para diferentes valores de número de muestras L. Pruebe por ejemplo los siguientes valores N=16, N=32, N=64, N=128. Indique a partir de que valor de N son distinguibles las tres frecuencias de la señal.

- ¿Cómo están relacionados los valores de L, N y la resolución en frecuencias?
- Compruebe que sucede en el espectro de la señal si una secuencia de L= 100 muestras de x[n] se rellena con ceros hasta N=128.
- Calcule la transformada inversa de la función X(w) utilizando la funcion ifft (X) para recuperar la señal en el dominio de tiempos.
- Suponga que se desea estudiar el contenido en frecuencias usando la FFT, de la siguiente señal.

x(t) = 0.0472*cos(2π(200)t + 1.5077) + 0.1362*cos(2π(400)t + 1.8769) + 0.4884 *cos(2π(500)t - 0.1852) +0.2942*cos(2π(1600)t -1.4488) + 0.1223*cos(2π(1700)t).

¿Cual es su frecuencia fundamental? ¿Que frecuencia de muestreo debe usarse? Estime un valor adecuado de N para obtener suficiente precisión en frecuencias. Represente |X(w)| y la fase de X(w) en función de w.

Ejercicio Nro. 2

Ud. labora para la división de equipos médicos de "SIEMENS" en el Dpto. de Optimización de Procesamiento Digital de ECG y en su primer día se le encomienda remover ciertas componentes del espectro ECG de un paciente sometido a Angioplastia Transluminal Percutanea en el laboratorio de Cateterismo. Se le proporciona un archivo ECG.MAT donde se encuentra un segmento ecg de 15000 muestras, el mismo fue obtenido a una frecuencia de muestreo Fs = 1000 muestras/seg. En un primer caso se requieren conservar las componentes hasta 50Hz y en un segundo caso es preciso conservarlas hasta los 300Hz.

Realice las siguientes actividades:

1. Diseñe los filtros necesarios (rechazabandas y pasabajo) utilizando **"fdatool"**
2. Exporte coeficientes y utilice las rutinas **"sosfilt".**
3. Grafique en una sola figura

   - ✓ Señal ecg proporcionada
   - ✓ Espectro señalando componentes indeseadas, utilice la rutina fftplot suministrada (recuerde modificar fftplot para no generar una nueva figura).
   - ✓ Señal ecg filtrada (para el Caso 1: pasan todas las frecuencias hasta 50Hz)
   - ✓ Espectro de señal ecg filtrada (caso 1)
   - ✓ Señal ecg filtrada (para el Caso 2: se requieren conservar componentes hasta los 300Hz)
   - ✓ Espectro de señal ecg filtrada (caso 2)

¿Cómo utilizar fdatool para hallar los coeficientes?

- En la línea de comandos de Matlab 7.0 escriba: **fdatool** y presione enter, obtendrá la siguiente interfase de diseño (fig.1).

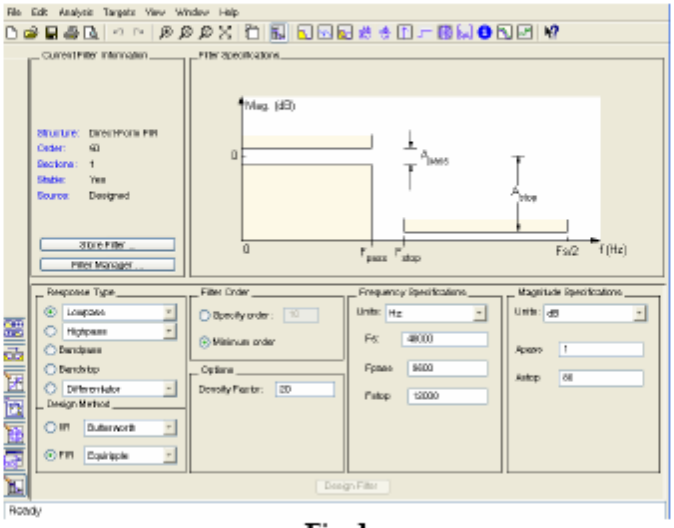


Fig. 1

- En el caso 1 Ud. tendrá que seleccionar un "Lowpass filter" (parte inferior izquierda de la interfase), tendrá que decidir si escoge un filtro tipo IIR o tipo

FIR, investigue que significa esto y que implicaciones tiene. Si elige un filtro tipo IIR, seleccione la opción "Butterworth", y si es tipo FIR: seleccione "Equirriple"; en ambos casos señale ventajas y desventajas.

- Para este ejemplo en específico se seleccionó: tipo: IIR-Butter (Lowpass) de orden mínimo para: fs = 1000 y magnitudes Apass =1 y Astop = 60. Presione el boton **"Desing filter"** al final y obtendrá una figura como la siguiente (fig.2):

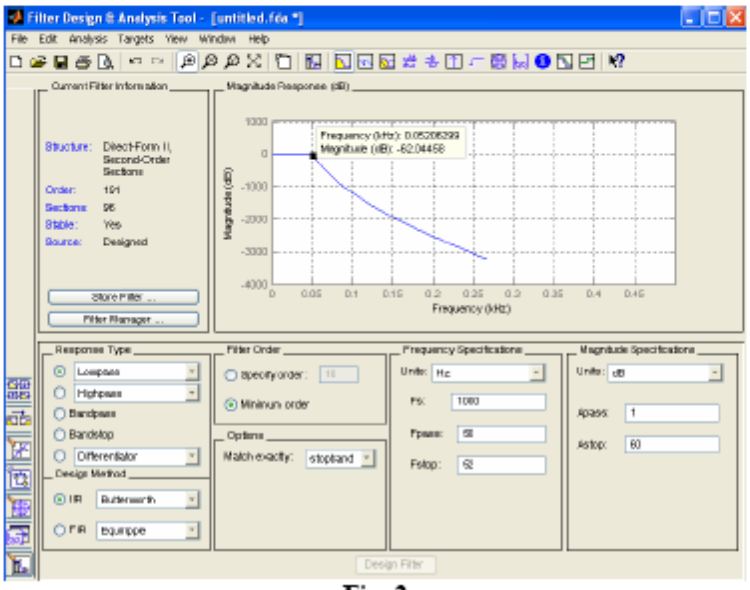


Fig. 2

- El siguiente paso es exportar los coeficientes: Vaya al menú **File->Export** y seleccione: export to: **"Workspace"** y export as: **"Coefficients"**, presione OK y tendrá los coeficientes en el "Workspace" como se muestra en la siguiente figura (fig.3): SOS es la matriz de coeficientes del filtro y G representa un vector de escalamiento para cada una de las secciones del filtro, aunque no la utilizaremos en ningún argumento.

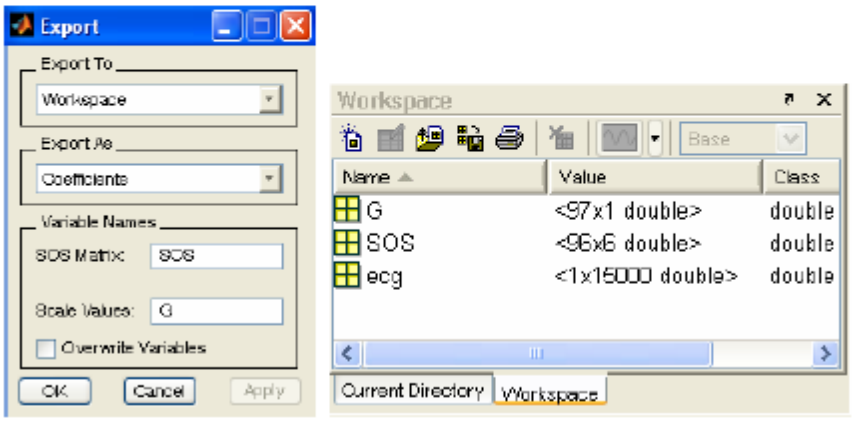


Fig. 3

- Coloque ecgfiltrada = sosfilt(SOS,ecg), grafique los espectros de ambas señales ("cruda" y filtrada), debería obtener una figura semejante a la siguiente(fig. 4):

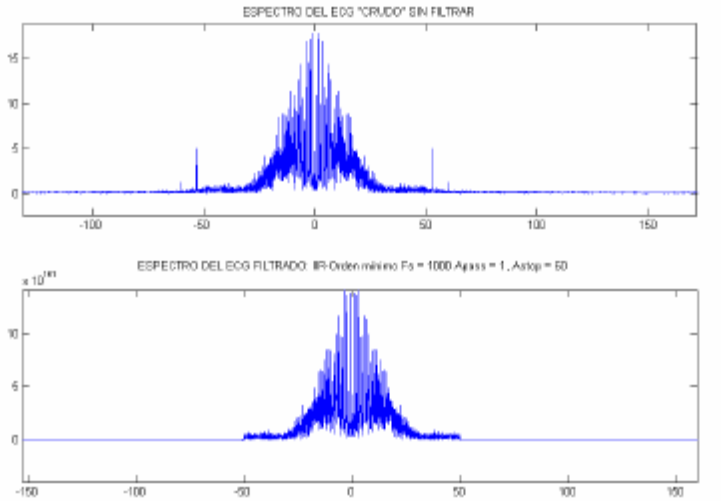


Fig. 4

- Ulitice un filtro butterworth del mismo orden anterior (191) pero esta vez programado sobre la línea de comando: [b a]=butter(191,wn) con wn=50/500=50/(fs/2), filtre la señal ecg con la función **filter y filtfilt**, guarde ambas señales, grafique con subplot en varios paneles: ecg+ecg(filter), ecg+ecg(filtfilt) y los espectros antes y después de filtrar (una vez). Si tiene problemas de aproximación numérica disminuya el orden del filtro. ¿Observando los ecg's en tiempo, que ventajas tiene filtfilt sobre filter?.

Ejercicio Nro. 3 (Convolución)

Haga una función para simular una convolución contínua en Matlab que tenga como variables de salida [y,ty] (ty, vector de tiempo de y) y como parámetros de entrada s1,t1,s2,t2,T; donde S1 y S2 son señales de entrada, t1 y t2 sus respectivos vectores de tiempo y T el tiempo de muestreo.