Universidad Simón Bolívar
Ingeniería Electrónica
SEÑALES Y SISTEMAS I

Esta guía se basa en el material desarrollado por el Prof. John Puentes (USB) y la Prof. Mary Díaz (USB). Ha sido revisado y ampliado por la profesora Trina Adrián de Pérez y el Prof. Noel Camilo Castro.

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)=et:

>>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