PROCESAMIENTO DIGITAL DE SEÑALES

PROFESOR: HUMBERTO LOAIZA C., Ph.D.

LABORATORIO No. 1. (Software)
Introducción General a Matlab – Generación de Scripts – Señales Básicas
Convolución – Interfaz gráfica

I. Objetivos

- *Efectuar una introducción general al ambiente de programación de Matlab y conocer las principales características de Matlab para el Procesamiento Digitales de Señales.*
- *Generar, visualizar y analizar señales en tiempo discreto empleadas frecuentemente en DSP.*
- *Implementar y analizar sistemas discretos en el dominio del tiempo mediante la convolución.*
- *Analizar las utilidades de la operación de deconvolución en sistemas de tiempo discreto.*

II. Introducción

Matlab (**mat**rix **lab**oratory) es una herramienta general de cómputo numérico que integra cálculo, visualización y programación en un ambiente interactivo donde los problemas y soluciones se expresan en forma muy similar a la notación matemática. Los arreglos constituyen el elemento base en su estructura de datos y no requieren ser dimensionados. Todos los datos se representan con doble precisión lo que hace más precisos los cálculos y la interacción más conveniente. Lo anterior implica que algunas veces se pueda requerir más memoria y mayor tiempo de procesamiento de lo realmente necesario.

Matlab dispone de una gran cantidad de algoritmos numéricos y opciones de visualización que vienen con el programa estándar. Las cajas de herramienta (toolbox) brindan un amplio y potente conjunto de funciones especializadas para diversas áreas del conocimiento. El toolbox de DSP ofrece una excelente funcionalidad para el procesamiento de señales tanto para el análisis de señales en el dominio del tiempo como en el frecuencial.

El ambiente interactivo de trabajo de Matlab favorece la manipulación y tratamiento de señales en forma dinámica por parte del programador. Permite la generación de funciones y de scripts que pueden combinarse con funciones propias de Matlab o con otras que hayan sido generadas por terceros programadores. Es permitido adicionar funciones que han sido escritas en C o Fortran. También, programas en C o Fortran pueden realizar llamados a Matlab y a las funciones de sus librerías.

Matlab no es un software de libre distribución. Su portal Internet se encuentra en http://www.mathworks.com. En este portal se puede consultar las ayudas de las funciones del programa, así como descargar archivos de libre distribución realizados por usuarios de Matlab.

1. Entorno de Trabajo con Matlab

Después de invocar el programa Matlab, aparece la interface gráfica de trabajo (desktop) con un conjunto de herramientas para manejar archivos, variables y aplicaciones asociadas con Matlab. Los componentes o herramientas que pueden aparecer normalmente en el desktop se indican en la figura 1.

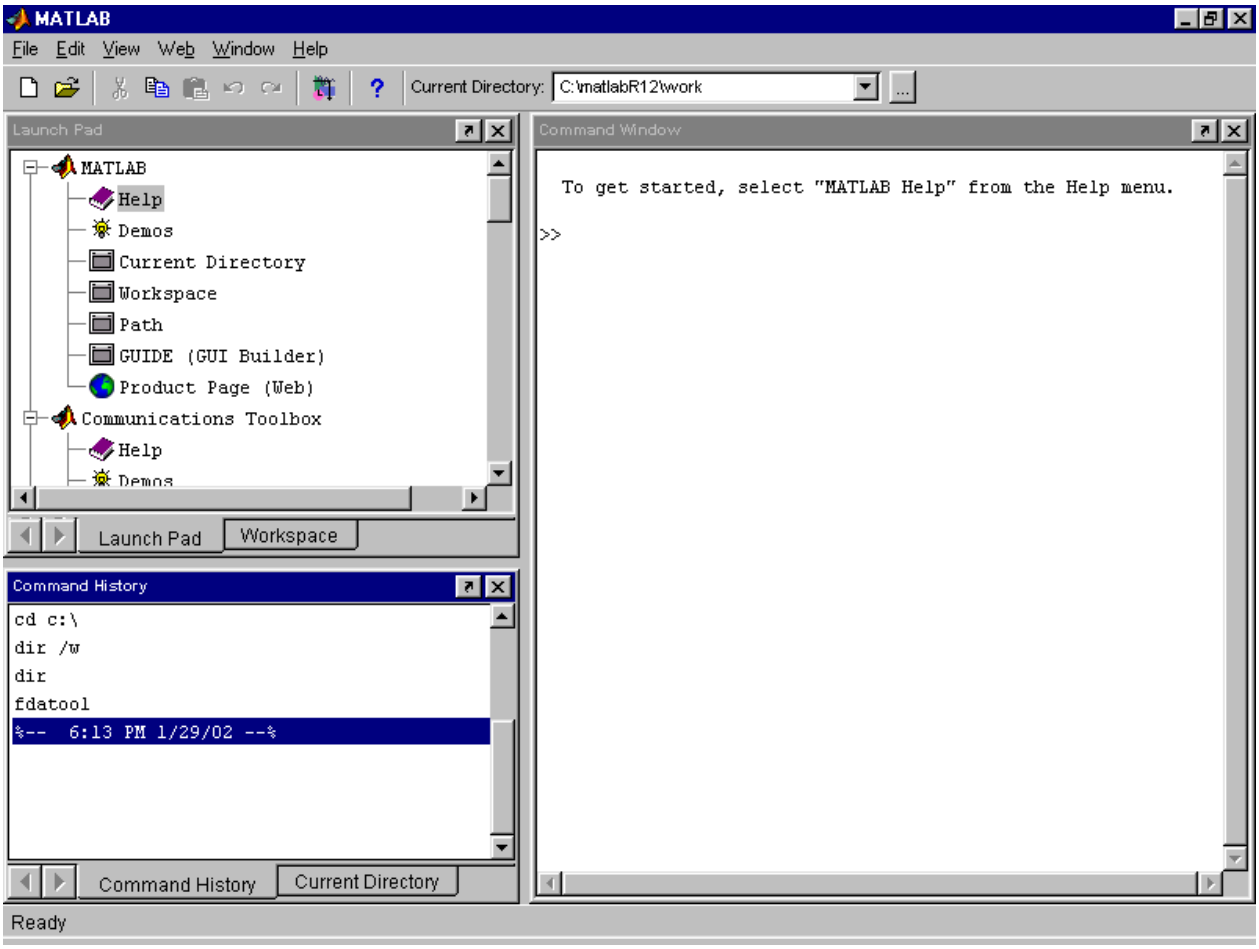

Figura 1. Interface gráfica de trabajo de Matlab.

Las herramientas que son manejadas en el desktop - aunque no siempre aparecen en el arranque pues depende de la configuración de inicio- se listan a continuación:

- ***Command Window.*** Corre funciones de MATLAB.
- ***Command History.*** Histórico de las últimas funciones entradas en la ventana de comandos, las cuales pueden ser copiadas y ejecutadas desde esta ventana.
- ***Launch Pad.*** Corre herramientas y accede a la documentación.

- ***Current Directory Browser.*** Visualiza archivos MATLAB y archivos relacionados; permite la realización de operaciones con archivos tales como abrir y encontrar contenidos.
- ***Help Browser.*** Permite la visualización y búsqueda de documentación de los productos de MATLAB.
- ***Workspace Browser.*** Permite la visualización y modificación del espacio de trabajo (características de variables utilizadas).
- ***Array Editor.*** Visualiza y edita el contenido de los arreglos en fomato de table
- ***Editor/Debugger.*** Sirve para crear, editar y depurar archivos .M.

Nota.
- Todas las funciones del desktop pueden invocarse desde la línea de comandos de la interface.
- Existen otras herramientas y funciones de manejo de figuras en ventanas que no se encuentran disponibles desde el desktop de Matlab.

1.1. Información sobre los comandos de Matlab : HELP

Para obtener una mayor información sobre las diferentes utilidades de Matlab se encuentra disponible el comando en línea **help**. El comando help seguido del nombre del comando visualizará en la ventana de comandos información sobre utilidad y modo de operación de la función. La ayuda generalmente se escribe dentro de los archivos .M de las funciones y está indicada con el caracter **%** al inicio de cada línea. La ayuda también puede invocarse mediante el botón help dispuesto en la barra superior del *desktop*, con lo que se obtiene una nueva ventana con varios menús y opciones para la búsqueda de la documentación requerida. Este modo es el recomendado mientras se familiariza con las herramientas de Matlab.

1.2. Arreglos (Array) en Matlab

Matlab trabaja solamente con un sólo tipo de objeto: el arreglo. Las variables, incluyendo escalares, vectores, matrices, cadenas, estructuras y objetos también son almacenados como arreglos. En C, los arreglos de Matlab se declaran del tipo *mxArray*, la cual es una estructura que contiene información detallada de las características de los datos.

1.2.1. Tipos de datos en Matlab

Matrices Complejas de Doble-Precisión

El tipo de dato más común en Matlab es la matrix compleja de doble precisión (no esparcida). Es de tipo *double* y presenta dimensiones de *m* filas por *n* columnas. Los datos son almacenados en dos vectores de números de doble precisión; uno contiene la parte real y el otro la parte imaginaria. Los punteros a estos datos se referencian como *pr* (pointer to real data) y *pi* (pointer to imaginary data), respectivamente. Una matriz real de doble precisión es aquella cuyo *pi* es nulo.

Matrices Numéricas

Matlab también soporta otros tipos de matrices numéricas. Estas son de punto-flotante simple-precisión y también enteros de 8, 16 y 32 bits, tanto signadas como sin signo. Los datos son almacenados en dos vectores de la misma manera que las matrices de doble-precisión.

Cadenas

Las cadenas son de tipo *char* y se almacenan de la misma manera que enteros de 16 bits sin signo, excepto que no contienen componente imaginario. Cada caracter en la cadena se almacena como un ASCII unicode de 16 bits. (A diferencia de C, las cadenas en Matlab no terminan con un caracter nulo).

Matrices esparcidas

Las matrices esparcidas tienen una convención de almacenamiento diferente a las matrices completas. Los parámetros *pr* y *pi* siguen siendo arreglos de números de doble precisión, pero se requiere de tres parámetros adicionales: *nzmax*, *ir* y *jc*.

Celdas de Arreglos (Cell Arrays)

Son colecciones de arreglos donde cada mxArray se referencia como una celda. Esto permite almacenar conjuntamente arreglos de diferentes tipos. Las celdas de arreglos son almacenados en forma similar a las matrices numéricas, exceptuando la parte que contiene un solo vector de punteros a los arreglos mxArray. Los miembros de este vector se denominan celdas. Las celdas pueden ser de cualquier tipo de dato soportado, aun otra celda de arreglos.

Estructuras

Una estructura de 1x1 se almacena de la misma manera que un arreglo de celdas de 1xn, donde n es el número de campos en la estructura. Los miembros del vector de datos se denominan campos. Cada campo está asociado con un nombre almacenado en el mxArray.

Objetos

Los objetos se almacenan y acceden de la misma forma que las estructuras. En Matlab, los objetos se denominan estructuras con métodos registrados. Fuera de Matlab, un objeto es una estructura que contiene almacenamiento para una clase-nombre adicional que identifica el nombre del objeto.

Arreglos Multidimensionales

Los arreglos pueden ser multidimensionales. Un vector utiliza para cada dato, espacios de almacenamiento iguales al tamaño del correspondiente tipo de dato. El almacenamiento de datos se hace de igual manera que en las matrices.

Arreglos Lógicos

Cualquier arreglo esparcido o numérico no-complejo puede ser marcado como lógico. El almacenamiento se efectúa de la misma forma que en arreglos no-lógicos.

Arreglos Vacíos

En Matlab, los arreglos de cualquier tipo pueden ser vacíos. Un mxArray es aquel que presenta al menos una dimensión igual a cero. Por ejemplo, un mxArray de tipo double, donde *m* y *n* son iguales a 0 y *pr* es NULL, constituye un arreglo vacío.

1.2.2. Matrices en Matlab

Una matriz es un arreglo bidimensional de números reales o complejos. El álgebra lineal define muchas de las operaciones con matrices que soporta Matlab, incluidas la aritmética de matrices, ecuaciones lineales, eigenvalores, valores singulares y factorización de matrices.

Creación de Matrices

Comúnmente el término matriz y arreglo se utilizan sin distinción. Más precisamente, una matriz es un arreglo rectangular de dos dimensiones de números reales o complejos que representa una *transformación lineal*. Las operaciones algebraicas lineales que se definen sobre matrices han encontrado aplicaciones en una amplia variedad de campos técnicos. (El paquete Symbolic Math Toolbox extiende las capacidades de Matlab a operaciones sobre varios tipos de matrices no-numéricas).

- *Creación directa*

La forma simple de crear una matriz es mediante la definición directa de sus elementos. Si se desea crear, por ejemplo, una matriz A de 4 filas x 3 columnas que contenga los números desde el 1 hasta el 12 .

```
>> A= [1  2   3 ; 4  5   6 ;  7  8   9 ;  10  11  12 ]
```

Observe que el final de una fila se indica con el símbolo punto y coma (;) y que las columnas se marcan con espacios en blanco entre los diferentes elementos de la fila. La matriz queda definida con los corchetes [ ].

- *Creación mediante funciones*

Matlab tiene un amplio número de funciones para crear diferentes clases de matrices. Las funciones *ones*, *zeros*, *magic*, *pascal*, *rand*, *sprand* son algunos ejemplos.

1.2.3. Vectores y Escalares

- Un vector columna es una matriz mx1. Ej.: u= [3 ; -8 ; 2.5]
- Un vector fila es una matriz 1x n Ej.: v= [2 8.2 –2.1]
- Un escalar es una matriz de 1x1 Ej.: s=7

1.2.4. Acceso a los elementos de una matriz o de un vector

Para acceder a cualquier elemento de una matriz o vector se utiliza el nombre de la matriz seguido por paréntesis que contienen los números de fila y columna del elemento, separados por coma. Los índices para acceder a éstos deben ser enteros positivos y ≥ 1.

2. Manejo y Generación de Señales

Las señales básicas que se utilizan en DSP son el impulso $\delta(n)$, las exponenciales complejas de la forma $a^n u(n)$, las ondas sinusoidales y su generalización a exponenciales complejas.

Puesto que el único tipo de dato numérico que maneja Matlab es la matriz MxN, las señales deben presentarse como vectores con longitud finita. Esto último contrasta con la solución analítica de muchos problemas donde una fórmula matemática puede representar señales de longitud infinita.

Otro aspecto a tener presente es la asociación de la posición de los elementos en los vectores a los índices del dominio temporal. Matlab asume por defecto los índices desde 1 a N, siendo N la longitud de dicho vector. De otro lado, un vector señal es el resultado de muestrear una señal sobre algún dominio donde los índices pueden ir desde 0 a N-1, o quizás desde –N a N. Por lo anterior, resulta claro que no se puede ligar la información sobre el dominio de muestreo con los valores del vector señal, lo que obliga a conservar por separado ambas informaciones.

Para la generación de señales resulta muy útil el uso de la notación vectorial de Matlab. Su notación de alto nivel para operar con vectores permite reducir el tiempo de elaboración y ejecución de programas. Por ejemplo, las buclas *for* son casi innecesarias. De igual forma, es preferible utilizar las funciones existentes y aplicarlas a los vectores directamente.

3. Archivos .M

Los archivos que contienen código del lenguaje de Matlab se denominan archivos M. Estos archivos pueden generarse con la ayuda de cualquier editor de texto, y pueden utilizarse como cualquier otra función o comando de Matlab. Existen dos clases de archivos M:

- **Scripts**. No aceptan argumentos de entrada ni retornan argumentos de salida. Estos archivos operan sobre los datos en el espacio de trabajo (workspace). Presentan gran utilidad en la automatización de un conjunto de pasos que necesitan ser ejecutados muchas veces desde la línea de comandos.

  Un script es un archivo .M que contiene una secuencia de instrucciones que son ejecutadas en forma secuencial cuando se invoca su nombre desde la ventana de comandos de Matlab.

Los scripts también pueden crear nuevos datos y producir salidas gráficas. Las variables que se crean se mantienen en el espacio de trabajo aún después de terminada la ejecución del script. Estos archivos pueden contener cualquier conjunto de comandos de Matlab, incluidos los comentarios. No requieren declaraciones o delimitadores de inicio o fin del script

- **Funciones**. Pueden aceptar argumentos de entrada y retornar argumentos de salida. Las variables internas son locales a la función. Utilizado preferiblemente cuando se desea expandir las librerías de funciones a una aplicación específica.

Los archivos M generados por el usuario pueden ser organizados en directorios y cajas de herramienta (toolbox) personales que pueden adicionarse a la ruta de búsqueda de Matlab.
Pueden existir archivos .M con el mismo nombre en diferentes directorios. En estos casos, Matlab ejecutará el primero que encuentre en la ruta de búsqueda.

Para visualizar el contenido de un archivo .M, se puede utilizar la función *type* seguida del nombre del archivo.

III. Funciones y comandos prácticos de Matlab

Para familiarizarse con el programa Matlab, verifique el funcionamiento de los comandos listados (en negrilla). Amplíe su conocimiento a través del comando *help*. Estos deben escribirse en la ventana de comandos.

- **exit**: finaliza la sesión de Matlab. Tiene la misma función que quit. Como alternativa puede seleccionarse EXIT MATLAB desde el menú File.
- **clc**: limpia la ventana de comandos.
- **…** : se utiliza para concatenar sentencias de comandos que requieren varias líneas.
- **dir**: visualiza los archivos contenidos en el directorio actual de trabajo.
- **exists**: verifica si una variable o archivo existe.
- **cd**: cambio de directorio de trabajo
- **what**: lista los archivos M, MAT, MEX, MDL y P. Puede listar igualmente los archivos dentro de subdirectorios del directorio actual.
- **which**: localiza funciones y archivos.
- **who, whos**: lista las variables creadas en el espacio de trabajo.
- **lookfor**: busca una palabra en todas las ayudas.
- **%**: caracter utilizado al inicio de toda línea de comentarios.
- **i, j**: unidad de los números imaginarios.
- **Inf** : retorna la representación aritmética IEEE del número infinito positivo.
- **NaN**: retorna la representación aritmética IEEE para un No-Número, que generan operaciones que no tienen definido resultados numéricos.
- **pi**: π, relación entre el perímetro y el diámetro de un círculo.

- *Funciones varias*

  - **keyboard**: comando para depurar programas. Detiene la ejecución de un archivo en la línea donde se encuentra esta instrucción, y da control al usuario desde el teclado para cambiar o examinar variables. Para salir de este modo, debe invocarse el comando *return*.
  - **close**: permite borrar las figuras visualizadas.
  - **input**: utilizada para entrar datos hacia un programa desde la línea de comandos.
  - **System**: ejecuta comandos del sistema operativo y retorna los resultados.
  - **dos** : ejecuta comandos del sistema tanto para Windows como para DOS.

- *Funciones para manejo de menús*

  - ***uimenu***: crea una jerarquía de menús y submenús que se visualizan sobre la barra de menú de ventana gráfica (figure) abierta.
  - ***Uicontrol***: crea objetos gráficos para implementar interfaces gráficas con el usuario (*uicontrol* = user interface controls). Matlab dispone de un amplio número de objetos uicontrol para diferentes propósitos, algunos de ellos son:
    Check boxes , Editable text, Frames, List boxes, Pop-up menus, Push buttons, Radio buttons, Sliders, Static text, Toggle buttons.
  - ***menu***: genera un menú de selección para el usuario, sobre una pequeña ventana.
  - ***uicontextmenu:*** crea el contexto de un menú, el cual es el que aparece cuando el usuario presiona el botón derecho del mouse sobre un objeto gráfico.

- *Herramienta de desarrollo de interfaces gráficas de usuario*

  - **G**raphical **U**ser **I**nterfaces **D**eveloping **E**nvironment ***GUIDE***. Invoca la herramienta de diseño de interfaces gráficas para usuario.

- *Cajas de diálogo predefinidas.*

  - ***dialog.*** Crea una caja de diálogo.
  - ***errordlg.*** Genera una caja de diálogo para enviar mensaje de error.
  - ***helpdlg.*** Crea una caja de diálogo de ayuda
  - ***inputdlg***. Genera una caja de diálogo para entrada de datos.
  - ***listdlg.*** Crea una caja de diálogo con una lista de selección
  - **msgbox.** Genera una caja de diálogo para mensajes
  - ***questdlg.*** Crea una caja de diálogo para preguntas
  - ***waitbar.*** Visualiza una barra de espera
  - ***warndlg.*** Genera una caja de diálogo de warning.
  - ***uigetfile***. Visualiza una ventana de diálogo para revisar un archivo en el directorio actual.
  - ***uiputfile***. Visualiza una ventana de diálogo para seleccionar un archivo para escritura en el directorio actual.

La gran mayoría de funciones de Matlab utilizan nombres/iniciales en inglés (ej: max, min, path,..) e instrucciones de lenguajes de programación (if , while, else, end,..), los cuales dan una idea general de la operación que ellas realizan. Lo anterior facilita la programación y también la búsqueda de funciones en la documentación.

IV. Procedimiento

1. Visualización de señales discretas

1.1. La representación gráfica de señales discretas se realiza generalmente con la función **stem()** de Matlab. El siguiente conjunto de instrucciones permite generar y graficar una onda senoidal en tiempo discreto de 30 muestras.

```
clear all;
n=-10:19;                    %generación del vector de instantes
seno = sin (n/2 + 10 );      %generación de la señal seno discreta
stem (seno);                 %graficar señal en tiempo discreto
xlabel ( 'instantes n' );
ylabel ( ' seno (n) ' );
```

1.2. Analice las instrucciones y determine la función de los principales comandos. Utilice el *help* para obtener mayor información.

1.3. Ejecute desde la *línea de comandos* de Matlab las instrucciones dadas en § 1.1. Analice la respuesta obtenida. Verifique si la numeración del eje horizontal corresponde con los valores de la señal seno.

1.4. Modifique el programa dado en §1.1. para corregir el problema detectado en el numeral anterior. Consigne los resultados.

2. Ejecutando un Script

2.1. Con ayuda de un editor de texto genere un archivo denominado **rang_mag.m** que contenga las instrucciones dadas. Matlab también suministra entre sus herramientas un editor, el cual puede ser invocado desde la barra de botones con la opción ***File***, ***New***, ***M-File*** o en forma más directa haciendo click sobre el ícono *página en blanco*. Verifique que el archivo quede grabado en su directorio de trabajo actual. (Es conveniente que cada usuario cree su propio archivo de trabajo, preferiblemente fuera del espacio de Matlab, o dentro del subdirectorio **work** de Matlab).

```
% Ejemplo de script: Nombre archivo: rang_mag  ---   Determina el rango de un cuadrado magico
clear all;
r = zeros(1,32);
for n = 3:32
  r(n) = rank(magic(n));
end
r
bar(r)
```

2.2. Determine qué hace el programa y analice la función de las principales instrucciones.

2.3. Desde la línea de comandos de Matlab invoque el script (nombre del archivo seguido de ENTER) para ser ejecutado. Consigne los resultados. Explique el porqué de la forma regular de las barras.

2.4. Coincide la respuesta de §2.2. con lo obtenido al ejecutar el script en §2.3? Qué significado tienen los valores de ***r***(***n***) ?

2.5. Verifique que después de finalizado el script las variables *r* y *n*, se mantienen en el espacio de trabajo. Para ello utilice la función ***whos*** y observe la ventana *workspace* de Matlab.

3. Generación de señales discretas

3.1. ***Señal senoidal***. Utilice las siguientes instrucciones para generar y visualizar una secuencia de datos discretos a partir de una señal senoidal. Escriba las instrucciones en un script y ejecútelo desde la línea de comandos de Matlab. Consigne los resultados.

```
clear all;
m=50;                   % Cantidad de muestras
frec=40;                % frecuencia de la señal en Hz
Tsample= 3/frec ;       % periodo de muestreo en segundos
n= -(m-1)/2: (m-1)/2;   % instantes de muestreo
nT= n* Tsample;
y=sin(2*pi*frec*nT);
plot(nT, y)
title('Señal Discreta');
xlabel ( 'nT')
ylabel ( 'Magnitud')
```

3.2. La función *plot* permite visualizar adecuadamente una señal seno *discreta*? Justifique su respuesta. Pruebe para Tsample=0.2/frec.

3.3. Repita las instrucciones anteriores pero en lugar de **plot ( )** invoque la funcion **stem( )**. Consigne y compare los resultados.

3.4. ***Señal sinc***. La señal sinc, con frecuencia angular w_c está definida como,

$$\operatorname{sin} c\,(n) = \begin{cases} \dfrac{w_c}{\pi} & n = 0 \\ \dfrac{w_c}{\pi}\,\dfrac{sen(w_c\ n)}{(w_c\ n)} & n \neq 0 \end{cases}$$

donde w_c es la frecuencia de corte de un filtro paso bajo asociado (la función sinc es la transformada inversa de Fourier de la respuesta en frecuencia en un filtro paso bajo)

Utilice las instrucciones que se dan a continuación para generar y visualizar una secuencia de muestras a partir de una señal sinc. Escriba las instrucciones en un script y ejecútelo desde la línea de comandos de Matlab. Consigne los resultados.

```
% Calcula 2m+1 muestras de la funcion sin(2*pi*fl*t)/(pi*t)
% para n=-m : m  (es decir, centrado alrededor del origen).
% fc  :Frecuencia de corte [Hz] asociada a un filtro paso bajo
%     :normalizada entre 0 y 1 respecto a la frecuencia de muestreo, por lo tanto la
%     :frecuencia maxima permitida es  fc=0.5.
% x   :salida,  muestras de la funcion sinc
```

```
clear all;
m=50;
fc=0.25;
wc=fc*2*pi;
n=(-m:m);      %2m+1 instantes de muestreo centrado alrededor de cero
xn=sin(wc * n);
xd=pi*n;

x(1:m)= xn(1:m)./ xd(1:m) ;
x(m+1)=2*fc  ;
x(m+2:2*m+1)= xn(m+2:2*m+1)./ xd(m+2:2*m+1) ;

% graficar señal
figure ;
stem(n, x);
title( strcat('Generacion de la Señal Sinc -- fc= ', num2str(fc) ) );
xlabel ('Instantes n');
ylabel ('Señal Sinc'),
```

Repita el numeral anterior para m= 5 y 150. Se presentan cambios en la señal resultante? Es irrelevante el valor de m ?

3.5. Repita §3.4 para fc=0.1 y 0.5 manteniendo m= 50. Se presentan cambios en la señal resultante? Es irrelevante el valor de fc? Explique qué sucede para fc>0.5.

3.6. ***Señal escalón***. Realice un script que genere señales escalón discretas de forma tal que puedan modificarse fácilmente su amplitud, duración e instante de inicio (desplazamiento en el eje de tiempo discreto). El script debe permitir la visualización de las señales generadas. Tenga cuidado de ajustar el eje horizontal para que la señal escalón no se visualice como una señal rectangular. Consigne el programa.

3.7. Pruebe el script realizado en el numeral anterior para valores diferentes de cada opción (amplitud, duración y desplazamiento). Consigne los resultados.

3.8. ***Señal rampa***. Realice un script que genere señales rampas discretas de forma tal que pueda modificarse fácilmente su pendiente, duración e *instante de cruce por cero* (desplazamiento en el eje de tiempo discreto). El script debe permitir la generación de valores positivos y/o negativos de amplitud en la rampa (desplazamiento en el eje de amplitud), así como la visualización de las señales generadas. Consigne el programa.

3.9. Pruebe el script realizado en el numeral anterior para valores diferentes de cada opción (pendiente, duración y desplazamiento). Consigne los resultados.

3.10. ***Señal exponencial (a^n para a real)***. Realice un script que genere señales exponenciales discretas de forma tal que puedan modificarse fácilmente su base, duración e instante de inicio (desplazamiento en el eje de tiempo discreto). El script debe permitir la visualización de las señales generadas. Consigne el programa.

3.11. Pruebe el script realizado en el numeral anterior para valores diferentes de cada opción (base positiva y negativa, duración y desplazamiento). Consigne los resultados.

3.12. ***Señal exponencial [(a+ j b)^n base compleja]***. Realice un script que genere señales exponenciales discretas de forma de forma tal que puedan modificarse fácilmente su base, duración e instante de inicio (desplazamiento a la izquierda y derecha del eje de tiempo discreto). El script debe permitir la visualización de las señales generadas tanto la parte real como la imaginaria como la magnitud y fase. Consigne el programa.

3.13. Pruebe el script realizado en el numeral anterior para valores diferentes de cada opción (base positiva y negativa, duración y desplazamiento). Consigne los resultados.

4. Interfaz gráfica: senales1

4.1. Realice un programa en Matlab (denomínelo **senales1**) que permita generar las señales discretas de §3.1, §3.4, §3.6, §3.8, §3.10 y §3.12 (adicionar el impulso unitario) para realizar manipulaciones sobre ellas. La operación a realizar así como los parámetros deben activarse con botones y menús. El programa debe permitir :

a. Visualizar las señales antes y después de la manipulación en el dominio del tiempo.
b. Realizar las operaciones: reflexión con respecto al origen de tiempos, desplazamiento, suma, multiplicación y escalado en amplitud de secuencias.
c. Descomponer cualquiera de las señales en sus componentes par e impar.

Consigne el código del programa y los resultados obtenidos. Explique el algoritmo utilizado.

5. Convolución

5.1. La convolución en el dominio discreto está definida por la siguiente expresión:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k) = \sum_{k=-\infty}^{\infty} h(k)\; x(n-k)$$

Realice un script en Matlab, denominado **conv_dis.m**, que permita calcular la convolución entre dos señales en tiempo discreto de acuerdo con alguna de las definiciones dadas arriba. *No debe utilizarse las funciones de Matlab para este fin* (conv, convn). El script deberá utilizar los siguientes parámetros:

x, *h*, vectores con las muestras de la señales a convolucionar
xno, instante cero en la secuencia *x*
hno, instante cero en la secuencia *h*
y, vector que contiene las muestras de la salida
yno, instante cero en la secuencia *y*.

El script también deberá visualizar en una misma ventana las gráficas de *x*(*n*), *h*(*n*), *y*(n) en zonas independientes, así como el nombre de cada gráfica. Para lograr esto, utilice la función **subplot( )** de Matlab. Consigne el programa y explique el funcionamiento del algoritmo.

Cuál será la longitud del vector resultante? Justifique su respuesta.

5.2. Si la respuesta impulsional de un sistema LTI es dada por h(n) y la señal de entrada por x(n), calcule la secuencia de salida *y(n)=h(n)*x(n)* del sistema mediante la utilización de el script **conv_dis** para las siguientes señales:

a- $h(n) = \{\underline{1}, 2, 3, 2, 1\}$ $\quad x(n) = \begin{cases} 1+n, & n=0,1 \\ 1-n, & n=2,3 \end{cases}$

b- $h(n) = \{\underline{1}, 2, 3, 2, 1\}$ $\quad x(n) = \{1, \underline{2}, -1, -2\}$

c- $h(n) = \{1, \underline{2}, -1, -2\}$ $\quad x(n) = \{\underline{1}, 2, 3, 2, 1\}$

d- $h(n) = \begin{cases} n+1, & n=0,1,2 \\ n-1, & n=3 \\ 1, & n=4 \end{cases}$ $\quad x(n) = \begin{cases} \lvert n\rvert, & n=-1 \\ 2, & n=0 \\ -2, & n=1,2 \end{cases}$

e- $h(n) = \{\underline{1}, 2, 3, 2, 1\}$ $\quad x(n) = \{1.5, \underline{2}, -2, -1\}$

5.3. Compare las respuestas obtenidas para cada caso en el punto anterior. Observe en qué instante se presenta el mayor valor de la salida $y(n)$. Explique claramente el resultado obtenido y confírmelo analíticamente.

5.4. Obtenga por convolución la respuesta de los sistemas **a** y **c** del numeral §5.2 ante la entrada seno calculada en §3.1, con frecuencia frec = 1Kz, 10KHz, 50 KHz, 100KHz. Fije la frecuencia de muestreo a 300 KHz para todos los casos. Ajuste el número de muestras **m** para que se pueda observar al menos dos periodos de la señal. Para cada caso grafique las señales de entrada y salida. Qué función puede atribuirse al sistema $h(n)$ en cada caso?

5.5. Repita §5.2 hasta §5.4 pero utilizando la función **conv( )** de Matlab. Indique claramente cúal es la diferencia en el procedimiento de cálculo entre esta función (algoritmo) y el script **conv_dis**. Existen diferencias entre los resultados obtenidos con las dos funciones? Compare los tiempos de ejecución con las funciones **tic** y **toc** de Matlab. Justifique sus respuestas.

5.6. Pruebe la función **deconv( )** de Matlab con las señales de §5.2 y los resultados de §5.5 Establezca la utilidad práctica de esta función. Indique igualmente como se podría establecer el instante n=0 en la secuencia de salida arrojada por deconv().

6. Consulta

6.1. Realice el resumen de un artículo que describa una aplicación de procesamiento digital de señales: Señales fisiológicas, señales sísmicas, señales sensoriales o aplicación en comunicaciones.
Consigne la referencia bibliográfica.

6.2. En la siguiente dirección se encuentra una amplia colección de diferentes señales fisiológicas: http://www.physionet.org/physiobank/database/

a. Seleccionar una base con patologías de una parte del cuerpo (p.ej. ECG).
b. Documentar la base de datos.
c. Graficar algunas de las señales de la base de datos seleccionada (mostrar señales de diferentes patologías) y clasificarlas según lo visto en clase.

Observación: conserve los programas desarrolladas durante la práctica, podrán servirle de apoyo en los laboratorios siguientes.

7. Informe

7.1. Presente un informe escrito claro en donde se consigne el procedimiento, los programas, las señales, las justificaciones de las respuestas y los resultados obtenidos. Igualmente incluya el *análisis* e *interpretación* de los resultados. También consigne las *conclusiones*, *observaciones* y la *literatura consultada*. Utilice en el informe la misma numeración de la guía de laboratorio.

7.2. Indique cual es el procedimiento para crear ayudas para los programas desarrolladas por el usuario y cómo pueden ser consultados a través del comando help (al igual que los comandos existentes). Desarrolle ayudas para cada programa implementado.

7.3. Consulte y explique cómo identifica Matlab los directorios donde se encuentran las funciones de los usuarios.

Notas: La omisión de alguno de los ítems en el informe representa una disminución de la nota.
El informe debe hacer referencia ordenada a cada uno de los puntos de la guía.
Para facilitar la sustentación ante el profesor, realice scripts donde se definan los datos y se invoquen las funciones desarrolladas.