Curso basico de MATLAB

1. ¿Para qué sirven las simulaciones?

En la actualidad hasta en los proyectos más elementales pueden utilizarse herramientas de simulación. Con la difusión de las computadoras se introdujeron programas de simulación basados en técnicas de cálculo numérico. Inicialmente, esta metodología se centró en instituciones universitarias o de investigación que pudieran disponer de los recursos necesarios. Un cambio de enfoque se produjo con la introducción y difusión de las computadoras personales, con lo cual los programas de simulación se hicieron universalmente accesibles de modo que prácticamente todo desarrollo de ingeniería sobre sistemas de control, aún siendo de los más simples, suele ser analizado mediante simulación para prever su desempeño. Paquetes como ***Matlab*** **y** ***Simulink*** son ampliamente utilizados para simular procesos.

2. ¿Qué es *Matlab*?

Matlab **(MATrix LABoratory)** es un entorno interactivo basado en matrices para la realización de cálculo numérico y visualización de resultados.

- Incorpora un lenguaje de programación que permite implementar programas complejos de modo relativamente simple.
- Permite la resolución de problemas sencillos sin escribir un programa y con facilidades de representación grafica de los resultados
- Es empleado para el desarrollo de cálculo numérico de propósito general pudiendo manipular vectores y matrices tanto reales como complejos con funciones y fórmulas de variadas ramas de la matemática y resolución de problemas con formulación matricial que aparecen en control, estadística y procesado de señales.

2.1 ¿Donde se utiliza?

Actualmente el sistema ***Matlab*** se utiliza:

- A nivel académico, dentro de la universidad.
- A nivel de investigación para la resolución de complicados problemas científicos.
- A nivel industria para la resolución de complicados problemas de ingeniería.

2.2 ¿Cuáles son los usos típicos?

- Matemática y computación.
- Desarrollo de algoritmos.
- Modelado, simulación y desarrollo de prototipos.
- Análisis de datos, exploración y visualización.
- Gráficos científicos e ingenieriles.
- Desarrollo de aplicaciones, incluyendo la construcción de interfaces gráficas para el usuario.

Matlab, como ya se ha dicho, esta orientado al cálculo numérico, a diferencia de otros software como Mathematica que está orientado al cálculo simbólico. Otra característica importante de ***Matlab*** es que esta orientado a vectores y matrices, es decir, las operaciones o funciones matemáticas que son válidas para números escalares también lo son para los vectores y matrices. Si por ejemplo V es un vector de 5 elementos, entonces cos(V) entregará un vector de 5 elementos con los valores de coseno para cada elemento del vector original.

2.3 ¿Cuáles son las características mas notables?

- ♦ La programación es mucho más sencilla.
- ♦ La exactitud de los números son mayores.
- ♦ Presenta una biblioteca matemática amplia.
- ♦ Presenta abundantes herramientas gráficas.
- ♦ Incluye funciones de interfaz gráfica con el usuario.
- ♦ Presenta capacidad de vincularse con lenguajes de programación clásicos.

2.4 ¿Qué son los toolboxes?

Matlab ofrece una familia de soluciones para aplicaciones específicas llamadas ***toolboxes***. Estos ***toolboxes*** son colecciones de funciones (archivos .m) que extienden el ambiente de ***Matlab*** para resolver algunos tipos de problemas particulares. Por ejemplo, se dispone *de* ***toolboxes*** para áreas tales como:
procesamiento de señales, diseño de sistemas de control, identificación de sistemas, simulación de sistemas dinámicos, optimización, redes neuronales, etc.

Entre los ***toolboxes*** más importantes se encuentran:

- **Curve fitting:** ajustes de modelos y análisis.
- **Data Acquisition:** adquiere y envía datos a un instrumento electrónico conectado al computador. (sólo para Windows)
- **Excel link:** permite usar ***Matlab*** con datos leídos directamente desde planillas Excel.
- **Image processing:** permite el procesamiento de imágenes, análisis y desarrollo de algoritmos.
- **Partial differential equation:** soluciona y analiza sistema de ecuaciones diferenciales parciales.

- **Signal Processing:** permite el procesamiento de señales, análisis y desarrollo de algoritmos.
- **Spline:** crea y manipula modelos de aproximación de datos Spline.
- **Statistics:** permite aplicar modelos estadísticos y modelos de probabilidades.
- **Structural Dynamics:** analiza modelos de elementos finitos y lleva a cabo análisis modales de sistemas mecánicos.
- **Wavelet:** analiza, comprime y saca el ruido de señales e imágenes usando técnicas de wavelet.

Con su amplio rango de herramientas para modelar sistemas de control, análisis, simulación y procesamiento de prototipos, ***Matlab*** es el sistema ideal para desarrollar sistemas avanzados de control.

- Se puede modelar un sistema de control usando los ***toolboxes*** para el diseño de controles avanzados: *Control System, Robust Control, μ-Analysis and Synthesis, Model Predictive Control, QTF Control Design y LMI control.*
- Posteriores análisis y refinamientos pueden ser efectuados estableciendo una simulación interactiva en ***Simulink,*** y luego sintonizar automáticamente los parámetros usando el *Nonlinear Control Design Blockset.*
- Finalmente, se puede generar código C para correr en controladores incrustados con *Real Time Workshop*.
- Combinando ***Matlab*** con *Signal Processing Toolbox, Wavelet Toolbox* y un conjunto de herramientas complementarias (tales como *Image Processing, Neural Network, Fuzzy Logic, Statistics* y otras) se puede crear un ambiente de análisis personalizado de señales y desarrollo de algoritmos DSP.
- Para simulación y desarrollo de prototipos se puede agregar ***Simulink*** y el *DSP Blockset* para modelar y simular sus sistemas DSP, y luego usar *Real-Time Workshop* para generar código C para su hardware designado.

3. ¿Qué es *Simulink*?

Simulink es un paquete de programas para planear, simular y analizar sistemas dinámicos. Tolera sistemas lineales y no lineales, en tiempo continuo, tiempo discreto o un híbrido de ambos. Los sistemas también pueden ser *monorate* o *multirate*.

3.1 ¿Cómo trabaja Simulink?

Para diseñar los modelos, proporciona una interfase gráfica para el usuario que permite construir modelos como diagramas en bloques, de la forma habitual en que se plantea el diseño de sistemas en el ámbito del control automático. Con esta interfase, se pueden dibujar los modelos así como si se hiciera con lápiz y papel (o como la mayoría de los libros de texto los ilustra). Esto es bastante diferente con respecto a los paquetes de simulación anteriores que exigen que se formulen ecuaciones diferenciales y ecuaciones de diferencia en un lenguaje o programa.

Simulink incluye una biblioteca de bloques que contiene entre otros: instrumentos de medición, fuentes, componentes lineales, no lineales y conectores. También se puede personalizar y crear bloques propios.
Cada modelo creado puede estar compuesto a su vez por subsistemas o niveles, por ello, los modelos tienen una organización jerárquica. En el ***Simulink*** se puede visualizar el sistema desde su nivel superior y bajar a través de los niveles con el propósito de ver la composición del mismo más detalladamente. Este acercamiento proporciona una mayor visión sobre cómo un modelo es organizado y cómo sus partes actúan recíprocamente.

Después de definir un modelo, se lo puede simular utilizando una variedad de métodos de integración. Usando bloques de graficación (instrumentos de medición), se pueden ver los resultados de la simulación tanto mientras es ejecutada como una vez que finaliza. Además, se pueden cambiar los parámetros de los bloques y de los subsistemas e inmediatamente ver la evolución de las variables. Los resultados de la simulación pueden ponerse en el *Workspace* de ***Matlab*** para el post-procesamiento y visualización.

Debido a que ***Matlab*** y ***Simulink*** están integrados, se pueden simular, analizar y revisar los modelos en cualquiera de éstos dos ambientes.

4. El entorno Matlab

Matlab interactúa con el usuario a través de ventanas. Las principales son las siguientes:

- *Ventana de comandos (Command Window)*: en ella se ejecutan las instrucciones.
- *Historial de comandos (Command History)*: historia de las instrucciones introducidas en la ventana de comandos.
- *Directorio actual (Current Directory)*: muestra los ficheros.
- *Espacio de trabajo (Workspace)*: muestra las variables almacenadas en memoria.
- *Ventana de figuras:* despliega las graficas que el usuario realice.

4.1 Manejo de comandos y funciones

Cada comando en ***Matlab*** es un archivo con extensión .m, por lo tanto es necesario tener las librerías en que se encuentran los comandos que se desean utilizar. Aunque la gran mayoría de los comandos utilizados siempre vienen incluidos en las librerías.

Matlab **NO** distingue entre mayúsculas y minúsculas en los comandos.

help	Documentación en línea. Nos permite ver lo que 'hace una función'
help general	Listado de comandos de propósito general
what	Listado de archivos con extensión **M, MAT, MDL y MEX** en el directorio corriente.

which	Localización de funciones y archivos.
demo	Ejecución de un demo de ***Matlab***.
pause	Hace una pausa en la ejecución.
quit ,exit	Para cerrar ***Matlab***.

Otros comandos son: ***type***, ***lookfor***, ***ver***, ***version***

4.2 Manejo de variables y workspace

who	Lista de variables actuales.
whos	Lista de variables actuales, permitiendo saber las características de la variable tal como su tamaño, el tipo o si es real o compleja.
load	Carga al workspace variables de disco.
save	Salvar variables del workspace a disco.
clear	Limpiar variables y funciones de memoria.
clc	Despeja la ventana de comandos
pwd	muestra cual es el directorio actual
size	Tamaño de la matriz
length	Numero de elementos de un vector
disp	Muestra los valores de una matriz o texto.
exist('c')	Chequea si la variable c existe

4.3 Comandos especiales

- El comando ***date*** proporciona la fecha en un formato simple:

```
>> date
ans =
09-Abr-2010
```

- El comando ***clock*** da como resultado de su ejecución: el año, mes, día, hora, minuto y segundo, en un formato de vector fila: [año, mes, día, hora, minuto, segundo]:

```
>> clock
ans =
1.0e+003 *
2.0100    0.0040    0.0090    0.0140    0.0160    0.0554
```

La misma información puede obtenerse utilizando un formato entero con ***fix(clock)*****:**

```
>> fix(clock)
ans =
2010        4        9       14       18       16
```

4.4 Sistema operativo y archivos

cd Cambia el directorio actual de trabajo.
Ejemplo: **cd c:\mb\work**

dir Muestra el directorio actual.
Ejemplo: **dir *.m** Lista todos los archivos de extensión M en el directorio actual

delete Borra un archivo u objeto gráfico.
Ejemplo: **delete *.p** Borra todos los archivos de extensión P desde el directorio actual

Es conveniente abrir una hoja en el editor de ***Matlab*** (M-file) para ir guardando lo que se va escribiendo. También es posible ir generando el programa directamente en el workspace de ***Matlab*** aunque la desventaja es que no se pueden guardar las instrucciones escritas.
Todos los comandos pueden utilizarse directamente desde la línea o prompt de comandos del ***Matlab*** (>>). Sin embargo la idea es hacer un archivo (con extensión .m) que contenga el programa que vaya realizando (para poder modificarlo, revisarlo, ejecutarlo otra vez) ya que es más ventajoso así.

4.5 Caracteres especiales

Para ***Matlab*** el carácter ***tanto por ciento (%)*** indica comienzo de comentario. Cuando aparece en una línea de comandos, el programa supone que todo lo que va desde ese carácter hasta el fin de la línea es un comentario.

% Precede a un comentario, las sentencias son ignoradas por el programa.

```
» % Este es un curso de Matlab
```

; Hace que la instrucción previa se ejecute pero no devuelve el resultado.

```
»y=2;
»z=2
z =
2
```

También se utiliza para definir una nueva fila de una matriz

```
»c=[1 2 4; 5 6 7]
c =
     1    2    4
     5    6    7
```

[ ] Se utilizan para ingresar valores o formar vectores y matrices

```
»a=[2 3 5]
a =
     2    3    5
```

( ) Define precedencia en expresiones aritméticas.
»(c+d)/2

También encierra argumentos de funciones en forma usual
»sin(a)

, Separa de elementos de una matriz
»s=[2,3,5]
s =
2 3 5

También se utiliza para separar argumentos enviados a funciones y declaración de funciones
»correlacion(signal1,signal2)
function
GRAFICAR(vector_valores,maximo_gramaje,minimo_gramaje)

= Se utiliza para asignar un valor a una variable
»a=[2 3 5]
a =
2 3 5

Algunos de los símbolos mas utilizados en ***Matlab*** se relacionan a continuación:

' Alt 39
(Alt 40
) Alt 41
. Alt 42
/ Alt 47
: Alt 58
; Alt 59
[Alt 91
\ Alt 92
] Alt 93
^ Alt 94

4.6 Lectura y escritura en archivos externos

Matlab permite salvar y recuperar datos de diferentes tipos de archivos, diferenciándolos básicamente por su extensión.

mat Archivo de datos binario. Se genera automáticamente con la instrucción **save** *archivo* y se recupera mediante la instrucción **load**. Hay que indicar que se pueden salvar los datos como caracteres ASCII empleando la opción *–ascii*, en cuyo caso el archivo no toma la extensión *.mat*.

dia Archivo ASCII que almacena las instrucciones ejecutadas durante una sesión de trabajo. Se genera automáticamente mediante la instrucción **diary**, pudiendo posteriormente convertirse en un *m-fichero*.

diary *nombrefichero* Eescribe los comandos entrados en el prompt y las salidas producidas de la sesión actual en el fichero *nombrefichero*. Se debe ejecutar al inicio de la sesión que se desea grabar.

» **diary c:\hola.txt** Guarda toda la sesión o mejor decirlo, todo lo que se ejecute a partir de esta instrucción en el archivo *hola.txt*. Antes de salir del ***Matlab*** es necesario dar la instrucción ***diary off*** con el fin de que el archivo sea cerrado y no se pierdan datos.

» **diary off/on** Al utilizar ***diary off*** después de haber iniciado una sesión, ***Matlab*** deja de guardar las instrucciones a partir de ese momento y al querer volver a grabar entonces lo activa utilizando ***diary on*** como se ilustra en el siguiente ejemplo.

En el siguiente ejemplo si el archivo *hola.txt* ya existe, ***Matlab*** anexa los datos al final del mismo.

```
» diary hola.txt
» y=2        % se asignó 2 a la variable y
» y
y =
     2
» diary off
» x=4        % se asignó 4 a la variable x
x =
     4
» diary on
```

Nota: La asignación de la variable x no queda registrada en el archivo *hola.txt*