UNIVERSIDAD NACIONAL EXPERIMENTAL DEL TACHIRA
DEPARTAMENTO DE INGENIERIA ELECTRONICA
LABORATORIO DE INSTRUMENTACION Y CONTROL
Mayo 2002

INTRODUCCION AL MATLAB. (PARTE 1).

Fundamentos

¿Qué es MATLAB? Es un ambiente de computación técnica de alto rendimiento, para cómputo numérico y visualización, que integra análisis numérico, cómputo de matrices, procesamiento de señales y gráficas en un ambiente fácil de usar, donde los problemas y sus soluciones se expresan como se haría en matemáticas, sin necesidad de la programación tradicional.

MATLAB, cuyo nombre es una contracción de Matrix Laboratory, es un sistema interactivo cuyo elemento básico es una matriz que no requiere dimensionamiento. Esta característica permite resolver muchos problemas numéricos en una fracción del tiempo que llevaría hacerlo en un lenguaje como Fortran, Basic, o C.

MATLAB también ofrece una familia de aplicaciones que se han dado en llamar Cajas de Herramientas o "Toolboxes". Un aspecto muy importante para la mayoría de los usuarios de MATLAB es que las toolboxes son colecciones de funciones, códigos o macros de MATLAB, conocidos como M-files, muy fáciles de comprender y de usar, que extienden el ambiente del programa con el fin de resolver clases particulares de problemas. Algunas de las áreas para las cuales existen toolboxes son: Procesamiento de Señales, Análisis y Diseño de Sistemas de Control, Simulación Dinámica de Sistemas, Identificación de Sistemas, Redes Neuronales, Ecuaciones Diferenciales Parciales, Procesamiento de Imágenes, Matemática Simbólica, Lógica Difusa, Estadística, Optimización, Comunicaciones, Finanzas, etc.

1.- ***Ventanas de Comandos, de Gráficos y de Edición***. Introducción al ambiente de MATLAB:

1.1 Desde el menú Inicio de Windows 98, abra MATLAB. Explore bajo la orientación del facilitador, los menús desplegables. No deje pasar ninguna duda.
1.2 Frente al "prompt" de MATLAB (») escriba **intro**, el programa responderá con un tour por algunas de las opciones disponibles. Observe las líneas de comando, las de comentarios y las respuestas del programa.

1.3 Continúe con **census**. Muestra algunas de las capacidades del programa para el ajuste de curvas y predicciones.

1.4 Escriba **spline2d**. En este ejemplo se muestra un método de entrada de datos por pantalla para construir una curva.

NOTA: hasta ahora habrá observado que todas las instrucciones escritas frente al prompt de MATLAB se han hecho en letras minúsculas. Esa es la opción predeterminada del programa. Si se usan letras mayúsculas, se producirá como respuesta un mensaje de error, a menos que cambie la opción predeterminada, lo cual no se recomienda. Los nombres de variables, vectores o matrices pueden escribirse en mayúsculas, minúsculas o combinaciones de ellas.

2.- Algunas instrucciones de uso general:

2.1 **help** seguido del nombre de una instrucción o comando de MATLAB muestra la ayuda en línea, que indica propósito, uso y formato de tal función. Por ejemplo
» help gcf.
Las funciones están divididas por categorías, si se desea saber, por ejemplo, qué instrucciones están relacionadas con funciones de matrices, escriba **help matfun** para obtener el listado.
MATLAB tiene varias categorías de funciones principales, además de las asociadas con cada una de las cajas de herramientas (toolbox). El comando de ayuda de MATLAB muestra una tabla de esas categorías. Escribiendo **help nombre_de_la_categoría** (e.g. **help control**) se obtiene una tabla de las funciones particulares asociadas a la categoría estudiada. Las diferentes categorías se enuncian a continuación:

general	Comandos de propósito general.
ops	Operadores y caracteres especiales.
lang	Lenguaje de programación: construcción y depuración.
elmat	Matrices elementales y manipulación de matrices.
elfun	Funciones matemáticas elementales.
specfun	Funciones matemáticas especiales.
matfun	Funciones de matrices – álgebra lineal numérica.
datafun	Análisis de datos y transformada de Fourier.
polyfun	Interpolación y polinomios.
funfun	Funciones de funciones y solución numérica de EDO.
sparfun	Matrices dispersas.
graph2d	Gráficos en dos dimensiones.
graph3d	Gráficos en tres dimensiones.
specgraph	Gráficos especializados.
graphics	Manipulación de gráficos.
uitools	Herramientas gráficas de interfase con el usuario.
strfun	Cadenas de caracteres.
Iofun	Entradas/salidas a archivos.
timefun	Tiempo y fechas.
datatypes	Tipos de datos y estructuras.
winfun	Archivos de interfase con el sistema operativo Windows (DDE/ActiveX).
demos	Ejemplos y demostraciones.
runtime	Kit de desarrollo del servidor Runtime de MATLAB.
rtw	Taller de tiempo real.
rtw\windows	Taller de tiempo real para Windows.
daq	Caja de herramientas para adquisición de datos.
daqdemos	Caja de herramientas para adquisición de datos - demostraciones.
dials	Conjunto de bloques de discos y medidores.
rptgenext	Simulink – Generador de reportes.
rptgen	MATLAB – Generador de reportes.
database	Caja de herramientas para bases de datos.
dbdemos	Caja de herramientas para bases de datos - demostraciones.
powerdemo	Power System Blockset Demos.

powersys	Conjunto de bloques para sistemas de potencia.
compiler	Compilador de MATLAB.
comm	Caja de herramientas de comunicaciones.
commmasks	Caja de herramientas de comunicaciones – Funciones de ayuda para enmascaramiento.
commsfun	Caja de herramientas de comunicaciones – Funciones S.
commsim	Caja de herramientas de comunicaciones – Archivos de Simulink.
symbolic	Caja de herramientas de matemáticas simbólicas.
nag	Caja de herramientas fundación NAG – Librería numérica y estadística.
nag\examples	Caja de herramientas fundación NAG – Librería numérica y estadística - ejemplos.
map	Caja de herramientas de mapeo.
mapdisp	Caja de herramientas de mapeo – Definiciones y despliegue de mapas.
mapproj	Caja de herramientas de mapeo – Proyecciones.
wavelet	Caja de herramientas de onditas.
wavedemo	Caja de herramientas de onditas - demostraciones.
pde	Caja de herramientas de ecuaciones diferenciales parciales.
finance	Caja de herramientas para finanzas.
calendar	Caja de herramientas para finanzas – Funciones de calendario.
findemos	Caja de herramientas para finanzas - demostraciones.
lmictrl	Caja de herramientas de control LMI
lmilab	Caja de herramientas de control LMI – Funciones lab.
qft	Caja de herramientas de control QFT.
qftdemos	Caja de herramientas de control QFT – demostraciones.
fixpoint	Conjunto de bloques para punto fijo.
fxpdemos	Conjunto de bloques para punto fijo - demostraciones.
dspblks	Conjuntos de bloques DSP.
dspdemos	Conjuntos de bloques DSP – demostraciones y ejemplos.
dspmasks	Conjuntos de bloques DSP – Funciones de ayuda para enmascaramiento.
fuzzy	Caja de herramientas de lógica difusa.
fuzdemos	Caja de herramientas de lógica difusa - demostraciones.
mpccmds	Caja de herramientas de control predictivo (MPC).
mpcdemos	Caja de herramientas de control predictivo – demostraciones.
fdident	Caja de herramientas de identificación en el dominio de la frecuencia.
fddemos	Caja de herramientas de identificación en el dominio de la frecuencia - demostraciones.
hosa	Caja de herramientas de análisis espectral de orden superior.
hosademo	Caja de herramientas de análisis espectral de orden superior - demostraciones.
stats	Caja de herramientas de estadísticas.
ncd	Conjunto de bloques para control nolineal
images	Caja de herramientas para el procesamiento de imágenes.
imdemos	Caja de herramientas para el procesamiento de imágenes - demostraciones e imágenes de muestra.
nnet	Caja de herramientas de redes neurales.
nndemos	Caja de herramientas de redes neurales - demostraciones.
nnutils	Caja de herramientas de redes neurales – utilidades.
commands	Caja de herramientas en análisis y síntesis Mu.
mutools\subs	Caja de herramientas en análisis y síntesis Mu – subrutinas.
signal	Caja de herramientas en procesamiento de señales.
siggui	Caja de herramientas en procesamiento de señales – Interfase gráfica con el usuario.
sigdemos	Caja de herramientas en procesamiento de señales – demostraciones.
splines	Caja de herramientas de interpolación segmentada (splines).
optim	Caja de herramientas de optimización.
robust	Caja de herramientas de control robusto.
ident	Caja de herramientas en identificación.
control	Caja de herramientas en control.
ctrlguis	Caja de herramientas en control- Interfase gráfica con el usuario.
sfdemos	Flujo de estados – demostraciones y ejemplos.
sb2sl	SystemBuild para el traductor de Simulink.
tour	Tour de MATLAB.
stateflow	Flujo de estados.
simulink	Simulink
blocks	Simulink – Librería de bloques.
simdemos	Simulink – demostraciones y ejemplos.
dee	Editor de ecuaciones diferenciales.
local	Preferencias.

2.2 Con **type** seguido del nombre de una instrucción se obtiene un archivo texto con información acerca de la instrucción (salvo que se trate de una función interna de MATLAB), incluyendo la forma como está programada. Ejemplo: **type tic**.

2.3 En algunos casos si el proceso de cómputo es demasiado largo o se hace infinito, es posible abortar la ejecución del programa utilizando **Control-C**.
2.4 Escribiendo **echo on** antes de ejecutar un archivo **.m**, permite ver las líneas de comando en la medida en que se van ejecutando. Se suspende con **echo off**.
2.5 El comando **tic** inicia un conteo de tiempo y **toc** lo finaliza, expresando en segundos el tiempo transcurrido entre los dos. Se puede usar, por ejemplo, para determinar el tiempo de ejecución de un programa, o para medir la duración de una sesión de trabajo.
2.6 Los comandos **who** y **whos** dan información acerca de las variables que se encuentran en el espacio de trabajo de MATLAB.
Cuando se utiliza el *Bloc de Notas* de *Microsoft Windows* para escribir y editar las variables, comandos, códigos, macros o funciones a utilizar, los archivos deben grabarse con la extensión **.m**. Si los archivos han sido grabados con la extensión **.m**, se podrán ejecutar desde la Ventana de Comandos con solo escribir el nombre del archivo (sin **.m**). Trabajando con *Windows 95* es imposible guardar un archivo como **.m** desde el *Bloc de Notas*, ya que *Windows* le agrega la terminación **.txt** . El Departamento de Asistencia Técnica de MathWorks sugiere el siguiente procedimiento para corregir la situación:

a) Abrir *Mi PC*, haciendo una doble pulsación sobre su icono.
b) Pulsar con el botón izquierdo del ratón sobre el menú *Ver*.
c) Seleccionar *Opciones*.
d) Seleccionar *Tipos de archivos*.
e) Seleccionar *Nuevo tipo*.
f) Escribir **M-file** en el campo de *Descripción del tipo*.
g) Escribir **.m** en el campo de *Extensión asociada*.
h) Pulsar en *Nuevo* para abrir la ventana de Acciones.
i) Escribir **open with Notepad** en el campo de *Acciones*.
j) Usar el botón *Examinar* para encontrar la ubicación del archivo *notepad.exe*.
k) Pulsar en *Aceptar* para cerrar la ventana de *Nueva acción*.
l) Pulsar en *Aceptar* para cerrar la ventana *Tipos de archivo*.

3 - Vectores y matrices:

3. 1 Los elementos de vectores y matrices pueden ser valores, expresiones, funciones y otros vectores o matrices. Ejecutar las siguientes instrucciones:

```
x = [-1.3   sqrt(3)      (1+2+3)*4/5      log(0)          log10(inf)   sin(0)   0]

length(x)

x(11)=pi

length(x)
```

3.2 Matrices grandes se pueden construir a partir de matrices más pequeñas.

Ejecutar los siguientes comandos:

```
a = [ 1   2  3  ;  4  5  6  ;  7  8  9]

r = [ 12  13  14]

b = [ a ; r ]
```

3.3 Se pueden hacer particiones de matrices utilizando los dos puntos (:). Analice el siguiente ejemplo y luego realice, al menos, otras cuatro particiones:

```
c = b(2:4,:)

d = b(2:4,2)

e = b(:)
```

3.4 Cuando una expresión es demasiado larga para escribirla en una sola línea, ésta puede escribirse fraccionada, en una línea se escribe parte de la expresión, después un espacio y tres puntos suspensivos (elipsis), se pulsa Enter y se continua en la siguiente línea.

```
s = [ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 …
5 5 5 5 5 5 5 5 5 5]
```

4.- **Archivos .m: "scripts", funciones y macros**: en su manera normal, MATLAB es un programa que opera en base a comandos o instrucciones. Al escribirse un comando, el programa produce una respuesta. MATLAB ejecuta secuencias de instrucciones almacenadas en archivos texto, que se encuentren grabados con la terminación **.m** (M-Files) en directorios que se encuentren en la ruta de acceso de MATLAB. Los archivos **.m** pueden ser de tres clases: **"scripts", funciones y macros**.

Los "scripts" automatizan una larga secuencia de instrucciones o comandos. Cuando se ejecuta un "script", MATLAB simplemente ejecuta, paso a paso, los comandos que se encuentran en el archivo, las instrucciones del "script" actúan globalmente sobre los datos en el espacio de trabajo. Los demos que se encuentran en MATLAB son buenos ejemplos de "scripts".
Los archivos de funciones permiten añadir nuevas funciones a las ya existentes. Gran parte de la potencia de MATLAB radica en su capacidad para crear nuevas funciones que resuelven problemas específicos de los usuarios. Un archivo **.m** de una función siempre comienza con la palabra **function**, a diferencia de un archivo de un "**script**", que puede comenzar con cualquier palabra o con líneas de comentarios. Los comandos evaluados por la función, así como las variables intermedias creadas por esos comandos son locales, operan al interior de la función y no actúan globalmente sobre el espacio de trabajo. Lo único visible en una función son la entradas y las salidas.

Los macros son pequeñas secuencias de instrucciones que pueden ser archivos .m o parte de ellos. Se crean como cadenas de texto y se ejecutan con el comando **eval**.

4.1 **Gráfico en MATLAB**: Para observar el potencial gráfico de MATLAB:
a) Abrir el Bloc de Notas y copiar el código que se lista a continuación:

```
close all
tmin=0;
tmax=2*pi;
dt=0.1;
t=[tmin:dt:tmax];
x=cos(t);
y=sin(t);
plot(x)
figure
plot(t,x)
figure
plot(t,x,':');
figure
plot(t,x,t,y);
figure
plot(t,x,t,y,’g’)
xlabel(‘tiempo’)
ylabel(‘x(t) & y(t)’)
title(‘Grafico de x(t) & y(t) versus t’)
legend('x(t)','y(t)')
grid
```

b) Grabar el programa como un archivo **.m**, con un nombre arbitrario que sea diferente a cualquiera de las funciones pre-definidas en MATLAB.
c) Escribir ese nombre frente al "prompt" de MATLAB y presionar ENTER.
d) Observar las figuras y comparar con el código del programa. De surgir alguna duda se debe usar la instrucción help seguida por la instrucción que se desea investigar o se debe consultar al facilitador.
e) Explorar las funciones loglog, semilogx, semilogy, axis, hold, figure, close, polar, bar, stairs.
f) Para incluir figuras de MATLAB en procesadores de texto como Word, se debe seleccionar la figura, presionar *Edit* en el menú de esa figura y presionar *Copy Figure* dentro del menú de *Edit*. Dentro de *Word* se selecciona la posición de la figura y se presiona *Control-V*.

UNIVERSIDAD NACIONAL EXPERIMENTAL DEL TACHIRA
DEPARTAMENTO DE INGENIERIA ELECTRONICA
LABORATORIO DE INSTRUMENTACION Y CONTROL
Mayo 2002

INTRODUCCION AL MATLAB. (PARTE 2).

Aplicaciones en Control

Polinomios:

Los vectores en términos de ciertas funciones matemáticas y de control pueden ser asociados con polinomios de orden decreciente, por lo tanto alguna de las operaciones típicas con polinomios se pueden efectuar de la siguiente manera:
Multiplicación: función **conv**
Ejemplo: Para multiplicar $\left(x^3 + 2x + 5\right)\left(3x^2 + 4x - 1\right)$
P1=[1 0 2 5]
P2=[3 4 –1]
P3=conv(P1,P2)

División: función **deconv**
Ejemplo: Para dividir P3 entre P1 (ejemplo anterior).
P4=deconv(P3,P1) (el resultado debe ser igual a P2).

Fracciones parciales: función **residue**
A partir de los polinomios del numerador y del denominador de una función de transferencia, ésta se puede descomponer en fracciones parciales con la instrucción:
[R,P,K] = residue(Num,Den), en donde R y P son los vectores columna que contienen los residuos y los polos, y K es el cociente.

Funciones de Transferencia:

Para escribir una función de transferencia y visualizarla en pantalla se introducen como vectores los polinomios del numerador y del denominador en la función **tf**:

Ejemplo:
Num=[2 0 1];
Den=[5 1 3 7];
G=tf(Num,Den)

Si en lugar de tener la versión extendida de la función de transferencia, se tienen los ceros, polos y ganancia, la función a usar es **zpk**:
Ejemplo:
Ceros=[-2.4 -3];

```
Polos=[-5 -1 -7 -2];
Ganancia=12;
H=zpk(Ceros,Polos,Ganancia)
```

Para convertir de la forma extendida a la forma zpk.

```
G2=zpk(G)
```

Para convertir de la forma zpk a la extendida.

```
H2=tf(H)
```

Para extraer información de una función de transferencia existente:
Vectores del numerador y denominador:

```
[N,D]=tfdata(G, 'v')
```

Ceros, polos y ganancia:

```
 [Z,P,K]=zpkdata(H, 'v')
```

Gráfico de los polos y ceros en el plano complejo s:

```
pzmap(G)
```

Una función de transferencia es coprima o irreducible cuando no tiene polos y ceros que se puedan cancelar. Si la función es reducible (no coprima), se pueden cancelar los términos comunes por medio de la instrucción minreal(Sys,Tol), en donde Sys es el sistema a estudiar y Tol es un parámetro opcional que indica la tolerancia para el nivel de cancelación. Ejemplos:

```
H3=minreal(H)        (sin tolerancia)
H4=minreal(H,0.1)
H5=minreal(H,0.2)
```

Diagramas de Bloques

Bloques en cascada o serie:

```
G1=zpk([-1],[-3 -5],1);
G2=zpk([-3],[-4 –6],3);
Gs=G1*G2
```

Bloques en paralelo:

```
Gp=G1+G2
```

Bloques en realimentación :

```
Gr=G1/(1+G1*G2)            (realimentación negativa)
```

Actividades: Tomar algunos ejemplos de diagramas de bloques y simplificarlos utilizando las instrucciones anteriores.

Respuesta en el Tiempo

MATLAB hace posible el análisis de la respuesta de un sistema representado por su función de transferencia a diferentes estímulos: impulso, escalón, rampa, etc. con instrucciones sencillas:

Respuesta al impulso:
[y1,t1]=impulse(H) (en este caso el vector tiempo es calculado automáticamente)
plot(t1,y1)

t2=0:0.01:6; (en este caso el vector tiempo es calculado por el usuario)
y2=impulse(H,t2)
plot(t2,y2)

Respuesta al escalón:
t3=0:0.01:10;
y3=step(H,t3)
plot(t3,y3)

Respuesta a una entrada arbitraria r(t) que tenga una transformada de Laplace sencilla:
En este caso:

a) Se obtiene R(s).
b) Se calcula Y(s)=G(s)*R(s)
c) Se calcula y=impulse(Y,t), donde t es el vector del tiempo definido por el usuario.

Ejemplo para una entrada rampa r(t)=5*t:
t=0:.1:3;
R=tf([5],[1 0 0])
Y=H*R
yr1=impulse(Y,t);
plot(t,yr1)

Respuesta a otras formas de excitación

a) Se define el vector del tiempo: t=tmin:dt:tmax;
b) Se define para este intervalo la función de excitación u como un vector del mismo tamaño del vector t.
c) Se calcula y=lsim(G,u,t).

Ejemplo para una entrada rampa r(t)=5*t:
t=0:.1:3;
u=5*t;
yr2=lsim(H,u,t);
plot(t,u,t,yr2)

Para una señal de ruido aleatorio:

```
t=0:.01:3;
u=rand(size(t));
ya=lsim(H,u,t);
plot(t,u,t,ya)
```

Para una señal senoidal:

```
t=0:.01:3;
u=sin(t);
ys=lsim(H,u,t);
plot(t,u,t,ys)
```

La respuesta de un sistema de segundo orden al escalón es muy importante en el análisis y diseño de sistemas de control. Considere la siguiente función:

$$F = K\frac{w^2}{s^2 + 2zws + w^2}$$

Escriba un programa que le permita dibujar la respuesta del sistema *F* a un escalón unitario si los parámetros son:

K	*w*	*z*
1	2	1
1	2	0.7
1	2	0.5
1	2	0.2
1	2	0
2	2	0.5
2	1	0.5
2	5	0.5
2	8	0.5

Para los casos anteriores:

a) Dibujar las curvas de respuesta y analizar los efectos de la variación de parámetros sobre:
   i. Tiempo de estabilización.
   ii. Tiempo de levante.
   iii. Sobrepaso.
   iv. Tiempo al valor máximo.

Notas:

1) Usar la función **subplot** para mostrar más de una gráfica en una misma figura.
2) Definir el vector tiempo: t=0:0.01:10;