Last modified: Tue Mar 13 09:42:53 ART 2007

# Señales y Sistemas (66.74)

# Práctica 0 : Introducción a MATLAB/OCTAVE

El objetivo de esta práctica es proveer al alumno con una breve guía sobre la utilización básica de MATLAB necesaria para el desarrollo de esta materia, mediante descripción de los aspectos básicos de manejo de este programa, funciones, archivos, etc. y una serie corta de ejercicios que cubrirían los aspectos más elementales expuestos. Una mas amplia cobertura de estos tema se puede encontrar en:

*The Student Edition of MATLAB*, The MATLAB Curriculum Series, Prentice Hall Eds.

También se puede utilizar el Octave como alternativa al MATLAB. El Octave es un paquete GNU inspirado en el paquete comercial MATLAB, con el cual es prácticamente compatible hasta las versiones 4. Está disponible en las máquinas de los laboratorios entrando en linux. Desde una xterm tipear ¨octave¨. Esto abrirá la ventana de comandos del programa en la misma terminal. El Octave está disponible para las distribuciones de linux más importantes. Depende del Gnuplot para poder representar gráficos. Al ser un software acogido a la GPL (licencia publica general) su código fuente esta disponible y puede ser ampliado y mejorado por cualquier usuario.

Manual:

http://www.octave.org/doc/octave_toc.html

http://www.cyc.ull.es/asignaturas/octave/ApuntesOctave/index.html

myprimer_tc.pdf

## Variables en MATLAB/OCTAVE

Los objetos con los que nos manejaremos principalmente en esta materia son los *escalares* y *vectores*. Para MATLAB/OCTAVE estos dos tipos serán simplemente un caso particular de *matrices* que son los objetos básicos en este lenguaje.

La definición de una variable escalar se realiza mediante la asignación de una constante o una expresión que involucre operaciones entre constantes u otras variables.

Ej.:

```
A = 20;

a = A/20 + 3
```

Los nombres de variables son sensibles a mayúsculas y minúsculas. En MATLAB/OCTAVE el punto y coma no tiene significado sintáctico como en PASCAL o C, sino que simplemente indica si la operación a realizar debe mostrar o no el resultado en pantalla.

Para definir una variable vector las ideas anteriores son aplicables, sin embargo hay que tener presente que un vector es una secuencia de números. La manera más simple de especificar esto es mediante la escritura de la secuencia, separando los elementos de ésta por blancos, entre corchetes:

Ej:

```
X = [1 2 3 4]
```

No hay diferencia entre los nombres de escalares y vectores, MATLAB/OCTAVE los distingue por el contexto. No es posible definir un *tipo* para las variables con en los lenguajes C o Pascal, porque para Matlab cualquier variable es siempre un double (8 bytes), excepto las strings, que son cadenas de caracteres.

Ej: la variable X anterior ocupa 32 bytes. En cambio la variable definida como

```
Y = 'hola';
```

solo ocupa 4 bytes. Es posible ver el listado de todas las variables definidas en el "workspace" de Matlab (o espacio de trabajo) con el comando `whos`. Es posible borrar una variable del "workspace" con el comando:

```
clear X
```

El comando `clear all` borra todas las variables presentes en el espacio de trabajo.

Tampoco es necesario inicializar los vectores con la dimensión que tendrán, puesto que el programa actualiza automáticamente la dimensión si esta se agranda:

Ej: después de ejecutar la sentencia anterior es válido hacer

```
X(5) = 5;
```

Referencia a elementos de un vector:

La instrucción anterior muestra además la manera de acceder a un elemento de un vector: el nombre del vector debe ser seguido de el índice del elemento en cuestión entre paréntesis. Observar que el primer elemento tiene un valor de índice igual a 1.

Secuencias: el operador dos-puntos

Existen operadores y funciones en MATLAB/OCTAVE que permiten definir secuencias de formas más amigables que este tipo de definición explícita: el *operador :* genera secuencias de números igualmente espaciados, ascendentes o descendentes:

Ej. 1:

```
Y = 1:5;
```

genera el mismo contenido que el del vector `X` del ejemplo anterior.

La sintaxis completa de este operador es

*inicio:paso:fin*

que genera una secuencia de números que comienza en *inicio,* se incrementa en *paso* hasta alcanzar el valor *fin.*

Ej. 2: compruebe qué secuencias se forman en los siguientes casos:

```
10:-1:1
```

```
1:-1:10
```

```
y = 0:pi/4:pi
```

Variables especiales

MATLAB/OCTAVE dispone de una serie de variables muy útiles para operaciones aritméticas: `pi, Inf, NaN, eps, i, j`

Probar las siguientes expresiones:

```
pi
```

```
eps
```

```
i
```

```
j
```

El significado de la variable `NaN` es Not a Number. La variable `eps` muestra la resolucion numerica de Matlab, mientras que las variables `i` y `j` toman por defecto el valor de sqrt(-1).

Existen funciones en MATLAB/OCTAVE apropiadas para definir casos comunes de vectores. Por ejemplo para definir un vector de 10 ceros:

```
a1 = zeros(1,10); % Define un vector fila de 10 elementos

a2 = zeros(10,1); % Define un vector columna de 10
elementos
```

(el símbolo `%` es utilizado como comentario). La instrucción `help` seguida del nombre de la función proporciona una descripción bastante completa de las funciones incluidas. Utilícela para entender el porqué de las definiciones anteriores, y para conocer la manera de usar estas otras funciones:

| | |
|---|---|
| `ones` | Vector de elementos 1 |
| `rand` | Vector de elementos aleatorios |
| `linspace` | Secuencia de N elementos linealmente espaciados |
| `logspace` | Secuencia de N elementos logarítmicamente espaciados |

Es posible combinar también secuencias de vectores para formar otros vectores mayores:

Ej:

```
x = [9:-1:1]

y = [0:10]

z = [y x y x]
```

Conociendo el funcionamiento del `operador :` podemos también acceder a más de un elemento del vector por vez:

Ej:

```
x = 10:10:50 % Da como respuesta: 10 20 30 40 50

x(1) % Da como respuesta: 10

x(1) = 0 % Da como respuesta: 0 20 30 40 50 (muestra todo
el vector)
```

```
x(1:4) = zeros(1,4) % Da como respuesta: 0 0 0 0 50
```

Ejercicios: Generar el vector que contenga las señales representadas en los gráficos siguientes:

## Operaciones entre vectores:

Las operaciones básicas entre vectores incluyen:

### Suma y Resta:

Se debe tener la precaución que los vectores sean de las mismas dimensiones. La excepción a esta regla es que se puede sumar o restar un *escalar* a todo un *vector*, lo que sería una simplificación a generar un vector constante de amplitud igual a la del escalar y luego sumarlo o restarlo.

Las funciones siguientes se usan para determinar las dimensiones de un vector, y son muy útiles a la hora de determinar el error cometido cuando hay dos vectores involucrados en una operación:

| | |
|---|---|
| `Size` | indica la cantidad de filas y columnas de una matriz |
| `Length` | indica la longitud de un vector |

### Multiplicación:

El operador `*` en MATLAB/OCTAVE tiene el sentido de multiplicación entre matrices. Entre vectores es posible efectuar este producto sólo entre un vector fila y otro columna, que tendrá el sentido de un producto escalar entre ambos, o que dará una matriz en el caso de multiplicar un vector columna por uno fila. Igual que en el caso de la suma y resta, es posible efectuar la multiplicación (o división) de un vector por un escalar, pero aquí tiene el significado usual.

Al utilizar MATLAB/OCTAVE en nuestra materia frecuentemente haremos la suposición de que un vector es la secuencia de los elementos de una función de variable discreta. Y puesto que los vectores serán utilizados como si fuesen funciones, nos interesará más la multiplicación elemento a elemento, es decir como si multiplicáramos funciones. Para esto se debe utilizar el operador `.*` (multiplicación precedido de punto). En general para cualquier operador es posible precederlo de punto, cambiándole el sentido a operador elemento a elemento. En este caso es necesario que las dimensiones de ambos vectores concuerden exactamente. Como ejemplo de la diferencia entre ambas operaciones mencionaremos que es posible hacer `x.^2` pero no `x^2` .

Operadores relacionales:

Es posible comparar dos vectores de iguales dimensiones o un vector contra un escalar, utilizando los siguientes operadores:

| | |
|---|---|
| < | menor |
| <= | menor o igual |
| > | mayor |
| >= | mayor o igual |
| == | igual |
| ~= | distinto |

El resultado es un vector de igual dimensión cuyos elementos son `1` donde la comparación es verdadera o 0 donde es falsa. Las comparaciones se efectúan elemento a elemento.

Existe una función que pueden utilizarse asociada a estas operaciones: la función `find(x)` devuelve como resultado un vector cuyos elementos serán los índices del vector `x` para los elementos distintos de cero.

Ej.:

```
i = find(x < 0);
```

```
x(i) = (-1)*x(i);
```

Las operaciones anteriores "rectifican" al vector `x`.

Operaciones con numeros complejos

Los numeros complejos están permitidos en todas las operaciones y funciones de Matlab.

Probar:

```
X = [1 2; 3 4] +i * [5 6 ; 7 8]
```

Nota: los valores de `i` y `j` pueden ser redefinidos, aunque en ese momento perderán su valor de `sqrt(-1)`

ademas el valor `sqrt(-1)` puede ser asignado a otra variable:

```
ii=sqrt(-1)
```

Otras operaciones aplicables a los vectores:

A continuación se presenta el listado de las funciones que más comúnmente utilizaremos en la materia:

| | |
|---|---|
| `Max` | valor máximo |
| `Min` | valor mínimo |
| `Sum` | suma de los elementos |
| `Prod` | Producto de los elementos |
| `Cumsum` | suma acumulativa de los elementos |
| `Cumprod` | Producto acumulativo de los elementos |

Listados de librerías de funciones:

Utilizando la instrucción `help help` se puede obtener un listado de todas las categorías de funciones de MATLAB. Por ejemplo si se ejecuta `help elfun` se obtiene el listado de las funciones elementales dsponibles en MATLAB. En OCTAVE mediante la instrucción `help -i` se ingresa en un help de texto pero interactivo (una "info page"). En dicha página los textos que aparecen marcados con "*" tienen referencias cruzadas, es decir que apretando sobre ellos, es posible ingresar en un submenú. En los submenús, la línea superior también tiene referencias cruzadas: apretando "P", "N" o "U" es posible

ejecutar Previous, Next o Up. Las categorías de OCTAVE son similares (pero no exactamente iguales) a las de MATLAB.

Ejercicios:

1. Hallar el valor medio de un vector. (También ver la función `mean`).
2. Hallar el desvío standard de un vector.
3. Generar una onda triangular a partir de una cuadrada (ayuda: utilizar `cumsum`).

## Graficación:

MATLAB/OCTAVE es muy amigable en este sentido. El siguiente es un listado de las funciones que más a menudo utilizaremos en la materia:

| | |
|---|---|
| `Plot` | Grafica los puntos de un vector uniéndolos con líneas |
| `Stem` | Grafica los puntos de un vector como señal discreta. En OCTAVE es otra opción del plot |
| `Loglog` | Idem `plot`, pero ambos ejes tendrán escalas logarítmicas |
| `Semilogx` | Eje x logarítmico, eje y lineal |
| `Semilogy` | Inverso del anterior |
| `axis` | Define los límites de los ejes |
| `figure` | Inicializa una nueva figura. Con un argumento numérico, se posiciona en una dada figura ya existente |
| `close` | Cierra una figura |

Cada una de estas funciones admiten 1 o 2 argumentos: en el caso de utilizar sólo un argumento, por ejemplo `plot(x)`, se grafican los elementos del vector en cuestión vs. sus índices. En un gráfico de este estilo el eje x siempre tendrá como primer elemento el 1. En cambio si se grafica `plot(t,x)` se graficarán los puntos correspondientes a los pares ordenados en `t` y `x`, pudiendo entonces representarse funciones para valores de `t<1`.

Ejercicio:

Grafique , para . Utilice pasos de *x* en el gráfico tales que la función aparezca lo más 'continua' posible.

Funciones adicionales para graficación:

- Títulos y nombres de ejes:

*Después* que se ha dibujado el gráfico es posible rotular las variables involucradas y darle un título general, mediante las funciones

`title`

`xlabel`

`ylabel`

En Octave hace falta un comando adicional para representar los titulos de los ejes sobre un grafico que ya está en pantalla: `replot`.

También es posible fijar el rango de los valores de los ejes con el que se muestra un dibujo mediante `axis`.

- Distintos tipos de trazos: es posible realizar gráficos con trazos distintos, como por ejemplo punteado, circulos, asteriscos, etc. Por ejemplo, `plot(t,x,'o')` graficará los puntos correspondientes a los pares ordenados en `t` y `x` como circulitos no unidos por ninguna línea. `help plot` proporciona más información sobre este tercer argumento opcional de la función.

Ejercicio:

Grafique las 8 raíces de la ecuación . (Ayuda: la función `roots` calcula las raíces de un polinomio).

- `Subplot`:

Es en general necesario mostrar más de un gráfico en la misma figura para su comparación. Esto se puede realizar mediante la función `subplot`. Esta instrucción, que debe ejecutarse *antes* de la graficación, nos permite crear una matriz de gráficos en la figura y posicionar el siguiente dibujo en uno de ellos.

Ejercicio:

Realice 4 gráficos en la misma figura de las funciones , con ; y

.

- Más de 1 figura:

Para que los gráficos se realicen cada vez en una figura diferente, antes de ejecutar las instrucciones correspondientes sedebe ejecutar la instrucción `figure`. La instrucción `close all` cierra todas las figuras existentes.

Mediante la funcion hold podremos superponer más de un trazado en la pantalla. Se desactiva con hold off.

## Archivos `.m` y `.mat`, y sentencias de control de flujo:

Cuando la complejidad de las instrucciones necesarias para realizar una operación aumenta, es mejor agruparlas en unidades lógicas o archivos ejecutables por MATLAB/OCTAVE como una única instrucción. Esto se logra mediante los archivos `*.m`, los cuales deben ser escritos mediante un editor de texto y salvados con un nombre y la extensión `.m`. El programa es invocado mediante el nombre con el que fue salvado (sin extensión).

**Nota 1**: En la barra de tareas de MATLAB la opción: File, New, M-file invoca a un editor de texto. En la versión 4.x, es el Notepad de Windows, mientras que en la versión 5.x el editor viene incorporado al Matlab.

**Nota 2**: Los archivos creados de esta manera sólo pueden ser invocados desde el directorio en el que fueron salvados. Es altamente recomendable organizar una estructura de directorios coherente, por ejemplo guardar todos los archivos `*.m` en un directorio `C:\usu\matlab`. Para que MATLAB tome este directorio como directorio actual de trabajo debe ejecutarse la instrucción `cd c:\usu\matlab .`

### Archivos `.m` y funciones

Es posible definir que un archivo que contiene una pieza de código de Matlab se considere una función si dicho codigo comienza con el encabezado `function`. En dicho encabezado estará además definida la sintaxis de la función, es decir, sus entradas y salidas. Por ejemplo la función siguiente:

```
function [suma] = sumar(x,y)
suma = x + y;
return;
```

devolverá la suma de dos numeros, los que se ingresan entre parentesis en la llamada a la funcion. Este código debe salvarse como `sumar.m` para poder ser ejecutado.

La diferencia entre un script que no es función y uno que sí lo es, está principalmente en que en las funciones las varibles son internas (no forman parte del workspace), y son automáticas (desaparecen después de ejecutada la función). En cambio un script que no

está encapsulado en una función deja los valores finales que toman las variables en el workspace al terminar.

Sentencias de control de flujo:

En este lenguaje se disponen de 3 estructuras básicas de control de flujo:

1. Ciclos `for`: La estructura básica de esta instrucción es:

   `for i =` *expresión*

   *sentencias;*

   `end`

   Ejercicio:

   Realice mediante esta estructura la definición de un vector `x` de 10 elementos todos ceros.

   **Nota:** El ejercicio anterior sería equivalente a ejecutar la instrucción única `x = zeros(1,10);` que es mucho más rápida que el ciclo propuesto en el ejercicio (por ejemplo probar el mismo ejercicio con 10000 elementos). Esto nos muestra que en lo posible hay que tratar de evitar la utilización de ciclos y estructuras si existe una función vecotrial ya predefinida que permite hacer lo mismo.

2. Ciclos `while`: La forma general es la siguiente:

   `while` *expresión*

   *sentencias*

   `end`

   Este tipo de ciclo es abierto, es decir el fin del lazo queda determinado en el momento de la ejecución de las sentencias correspondientes.

   Ejercicio:

   Calcular la función mediante su desarrollo en serie ,
   sumando términos mientras cada término no sea menor que . Ingrese el punto en el cual realizar el cálculo mediante la instrucción `input`.

3. Sentencias `if`: Siguen la siguiente estructura:

   `if` *expresión1*

   *sentencias1*

   `elseif` *expresión2*

   *sentencias2*

   ...

   `else`

   *sentencias3*

   `end`

   Los puntos suspensivos significan que las sentencias `elseif` pueden repetirse tantas veces como sea necesario. Estas sentencias o el `else` pueden estar ausentes, pero no el `end` final.

Instrucciones `save y load`:

Es posible despues de haber generado una variable salvar su valor en un archivo para volver a cargarlo más tarde:

```
save tmp x
```

guarda la variable `x` en un archivo llamado `tmp.mat`. Si no se indica nombre de variables a continuación del nombre del archivo, todas las variables son salvadas.Con

```
load tmp
```

se vuelve a recuperar la información que fue salvada en `tmp.mat`.

Los archivos de datos (.mat en MATLAB) en OCTAVE tienen la extension .oct y son archivos de texto.

Algunos comandos generales:

A continuación se listan una serie de comandos generales útiles. Los tres primeros conforman las herramientas básicas para seguir aprendiendo más sobre Matlab. Los últimos son necesarios para la ubicación de los archivos y de los directorios de trabajo, etc.

| | |
|---|---|
| `lookfor` | Busca todas las funciones que contienen una cierta palabra |
| `help` | Breve descripcion de uso de una funcion. |
| `which` | Indica el directorio donde se encuentra una función |
| `cd` | Cambia de directorio |
| `dir (ls)` | Lista el contenido de un directorio |
| `pwd` | Da el nombre del directorio corriente |
| `path` | Listado de todos los directorios que estánen el path de Matlab |

Directorio de trabajo en linux desde octave:

Algunos comandos del shell de Unix están implementados en OCTAVE/MATLAB. El comando `pwd` indica el path completo del directorio actual de trabajo; `ls` muestra un listado de todos los ficheros de ese directorio; y `cd ..` cambia al directorio inmediatamente superior al actual (u otro cualquiera que se especifique en lugar de los dos puntos). Hay que tener en cuenta que el directorio actual para OCTAVE/MATLAB será aquel desde el cual se invocó octave si no se ha hecho ninguna operación de cambio de directorios.

## Sitio web de Señales y sistemas:

Durante el transcurso de la materia varios tipos de datos necesarios para realizar los trabajos prácticos, o informativos (como los textos de estas guías, parciales y coloquios anteriores) están disponibles en:

**http://www.fi.uba.ar/materias/6607**

Se recomienda consultarlo periódicamente para enterarse noticias, modificaciones de calendario, últimas actualizaciones de las guías, notas de parciales y finales, etc, etc, etc.