CÁTEDRA DE MÉTODOS NUMÉRICOS

Ingeniería Biomédica y Licenciatura en Matemática

Año 2011

MÉTODOS NUMÉRICOS Y MATLAB

INTRODUCCION

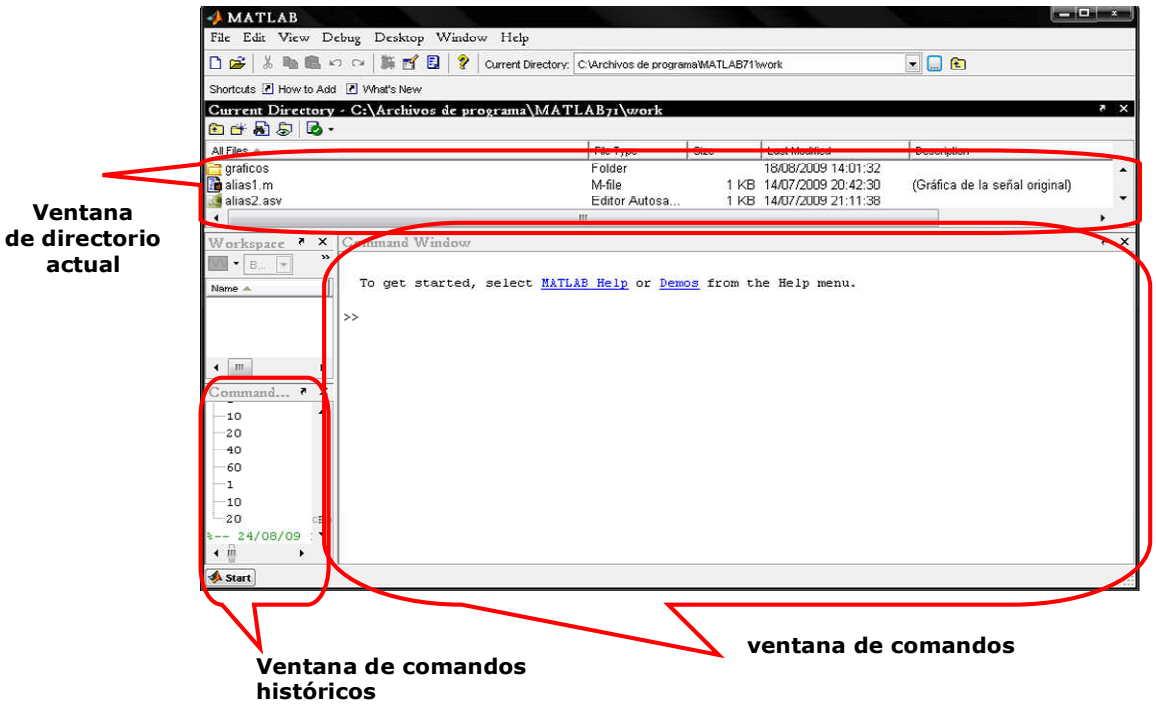
MATLAB (laboratorio de matrices) es un software matemático muy versátil que presenta un entorno interactivo y un lenguaje de programación para cálculos científicos y técnicos que permite realizar cálculos numéricos, análisis de datos y gráficos. Estas características hacen que MATLAB sea muy usado en la implementación de métodos numéricos, en particular para aplicaciones de ingeniería.
Los distintos usos que se le pueden dar son:

- Matemáticas y cálculos
- Desarrollo de algoritmos
- Modelado y simulación
- Análisis y visualización de datos
- Gráficos científicos e ingenieriles
- Desarrollos aplicados, incluyendo la construcción de interfaces gráficas.

Hay dos entornos básicos en MATLAB, un entorno interactivo y un entorno de programación mediante el uso del editor medit.

ENTORNO INTERACTIVO

MATLAB posee un entorno interactivo visual donde el cursor o prompt (caracter **»**) indica la línea donde se puede ingresar un comando. A continuación se hará una breve reseña de cómo usar este entorno.



Algunas consideraciones generales previas:

- ✓ MATLAB distingue entre mayúsculas y minúsculas.
- ✓ Los comentarios deben ir precedidos por %, lo que es lo mismo, MATLAB ignora todo lo que vaya precedido por el símbolo %.

Se pueden hacer cálculos matemáticos directos, como si usáramos una calculadora, sin necesidad de ser asignados a una variable en concreto. Por defecto, el resultado de una operación matemática o lógica se almacena en una variable de *"respuesta por defecto"* llamada ans. Por ejemplo:

```
»2+3
ans =
5
```

Sin embargo, si el cálculo se asigna a una variable, el resultado queda guardado en ella:

```
»x=2+3
x =
5
»x=2+3;  % el punto y coma evita que se muestre el resultado del cálculo
```

VARIABLES CON VALORES PREDETERMINADOS

MATLAB tiene la ventaja de poseer variables con valores predeterminados, esto es, que tienen un valor por defecto.
La variable pi, por defecto posee el valor

```
» pi
ans =
   3.14159265358979
```

```
»eps % Épsilon de la máquina. Obsérvese que MATLAB trabaja en doble precisión
ans  =
2.2204e-016
```

A pesar de tener estas variables valores predeterminados siguen conservando la propiedad de ser *"variables"*, es decir, se les puede asignar otros valores. Por ejemplo:

```
»eps=7
eps =
7
```

En este caso el valor predeterminado se perdió y es reemplazado por el número 7. Para reestablecer el valor por defecto deberá cerrar la sesión abierta e iniciar una nueva.

FUNCIONES PREDEFINIDAS

MATLAB posee un gran repertorio de funciones matemáticas, lógicas y otras específicas en forma de ToolBoxes o herramientas. Algunas de las más usadas se exponen a continuación:

Función coseno:

```
% pi es una variable con valor predeterminado 3.14159...
»cos(pi)
```

Función exponencial:
% Función exponencial evaluada en 1, es decir, el número e
»exp(1)

Función valor absoluto:
»x=abs(-6);

MATLAB trabaja con aritmética compleja. La unidad imaginaria se representa en MATLAB como i o j, variables con dicho valor como predeterminado:
»sqrt(-4)
ans =
0+ 2.0000i

Existen numerosas funciones específicas para el manejo de matrices. Para ver un listado de ellas se recomienda visitar el help (ayuda) que posee MATLAB.

FORMATO

El usuario puede controlar el número de decimales con que aparece en pantalla el valor de las variables, sin olvidar que ello no está relacionado con la precisión con la que se hacen los cálculos, sino con el aspecto con que éstos se muestran:
»1/3
ans =
0.3333
»format long
»1/3
ans =
0.33333333333333
»format % Vuelve al formato estándar que es el de 4 cifras decimales

VARIABLES

No es necesario declarar las variables, basta con darles un valor. Pero es importante tener claro qué variables se están usando porque es fácil cometer errores. Por ejemplo, se puede reutilizar la variable **pi** asignándole un número pero con ello se pierde el valor predeterminado 3.14159.

Otra consecuencia de no declarar variables es que al momento de la asignación se establece automáticamente su tipo de dato. Así, si inicialmente realizamos la operación x=2; el tipo de dato de la variable es entero pero si luego ejecutamos x='Hola Mundo'; la variable cambió de tipo a string (cadena de caracteres).

Nota: Todo lo que es de tipo carácter o cadena de caracteres va encerrado entre comillas simples (').

Todas las variables son matrices que no tienen dimensión estática sino que ésta se determina una vez usada dicha variable. Si se realiza la asignación de un solo valor a una variable el sistema la interpreta como una matriz de 1 fila y 1 columna.

Algunos ejemplos:

Para definir una variable de nombre x que contenga el valor 4:

```
»x=4
x=
4
```

Para definir un vector fila, basta introducir sus coordenadas entre corchetes:

```
»v=[1 2 3] % Vector de 3 coordenadas
v =
1 2 3
```

Si queremos declarar un vector de coordenadas equiespaciadas entre dos dadas, por ejemplo, que la primera valga 0, la última 20 y la distancia entre coordenadas sea 2, basta poner:

```
»vect1=0:2:20
vect1 = 0 2 4 6 8 10 12 14 16 18 20
```

Las matrices se escriben como los vectores, pero separando las filas mediante un punto y coma; así una matriz 3x3 se escribe como:

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

```
»v=[1 2 3];
»w=[4 5 6];
»mat=[v;w;0 0 1] % También es una matriz 3x3
mat =
1 2 3
4 5 6
0 0 1
```

A los elementos de una matriz se accede escribiendo el nombre de la matriz y, entre paréntesis, los respectivos índices:

```
»mat(1,3) % Elemento en la primera fila y tercera columna de la matriz mat
ans =
3
```

También se puede acceder a una fila o columna completas,

```
»mat(:,2) % Segunda columna de mat

ans =
2
5
0
```

```
»mat(2,:) % Su segunda fila
ans =
4 5 6
```

OPERACIONES CON MATRICES

Las operaciones de suma y resta de matrices se realiza usando simplemente los operadores correspondientes (+) y (-).

Para el caso de la multiplicación de matrices se usa el operador *. Para ello las matrices deben ser cuadradas o cumplir con que la cantidad de columnas de la primera debe ser igual a la cantidad de filas de la segunda matriz.

Muchas veces se desea elevar al cuadrado cada elemento de un vector, y es aquí donde se debe tener cuidado porque si se usa el operador * o el operador ^ (para elevar al cuadrado) MATLAB intentará realizar una multiplicación de matrices.
Ejemplo 1:

```
» x=[1 2 3 ];
» x*x
??? Error using ==> *
Inner matrix dimensions must agree.

» x^2
??? Error using ==> ^
Matrix must be square.
```

Para salvar este problema se debe anteceder los símbolos (*) y (^) con un punto.

```
» x.*x
ans =
   1    4    9

» x.^2
ans =
   1    4    9
```

Como se puede observar la respuesta es un vector cuyos elementos son el resultado de elevar al cuadrado cada elemento del vector x.

GRAFICACIÓN

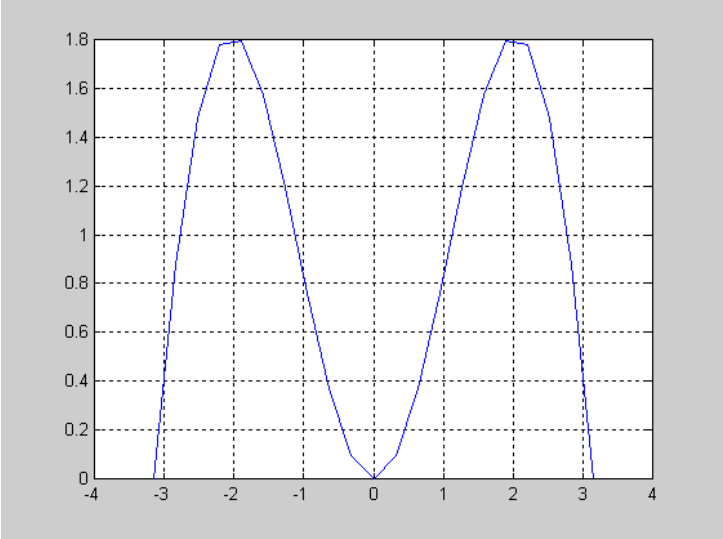
Las posibilidades de realizar y manipular gráficos es una de las grandes ventajas del uso de MATLAB. Para ello cuenta con una importante cantidad de comandos de graficación.

El comando plot permite graficar funciones. Como argumento/s plot recibe la/s función/es en forma de vector/es y genera en una ventana independiente un gráfico a partir de los datos.

Ejemplo:

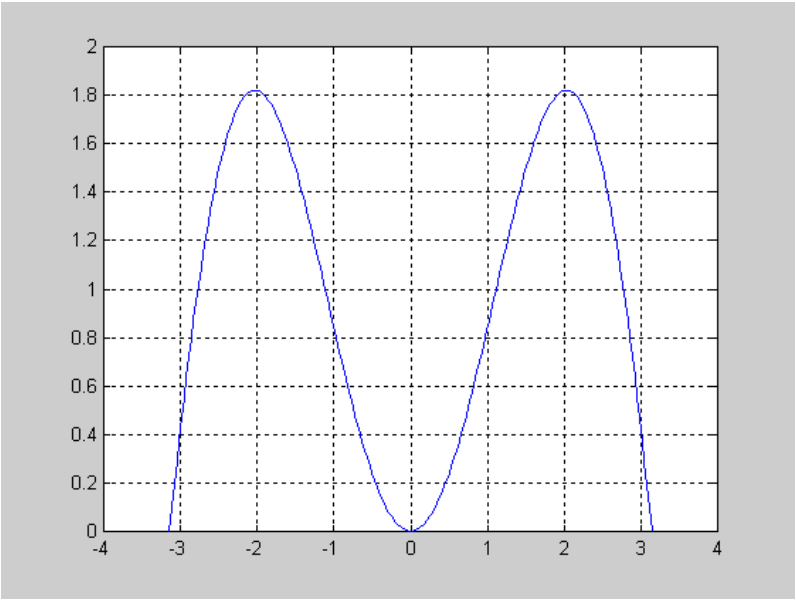
```
»x=pi*(-1:0.1:1); %x es un vector con valores equiespaciados múltiplos de pi
»y=x.*sin(x);
```

% y es un vector que resulta de aplicar la función x*seno(x) a cada elem. de x
»plot(x,y) % Por defecto une los puntos (x(i),y(i)) mediante una poligonal
» grid % dibuja una cuadrícula
El resultado se muestra en una nueva ventana:



Obs: Siempre se trabaja con pares ordenados (x_i, y_i), x_i pertenece al vector x, e y pertenece al vector y_i. Esto significa que a mayor cantidad de puntos mejor aproximación gráfica se puede tener. Con este criterio se podría mejorar el gráfico anterior de la siguiente manera:

```
»x=pi*(-1:0.01:1);
»y=x.*sin(x);
»plot(x,y);
» grid;
```



También pueden graficar funciones en un mismo eje coordenado. Así:

»plot('sin(x)',[0 2*pi]) % Dibuja la función seno en el intervalo [0,2*pi]
»hold on % Mantiene en la ventana gráfica los dibujos anteriores
»plot('cos(x)',[0 2*pi]) % Dibuja sobre la gráfica anterior la función
»hold off % Con esto olvida los dibujos anteriores

El resultado es una sola gráfica con dos curvas.

Otra forma de graficar varias funciones en un solo gráfico es:
»plot (x1,y1, x2,y2, x3,y3,... xn,yn)

ENTORNO DE PROGRAMACION

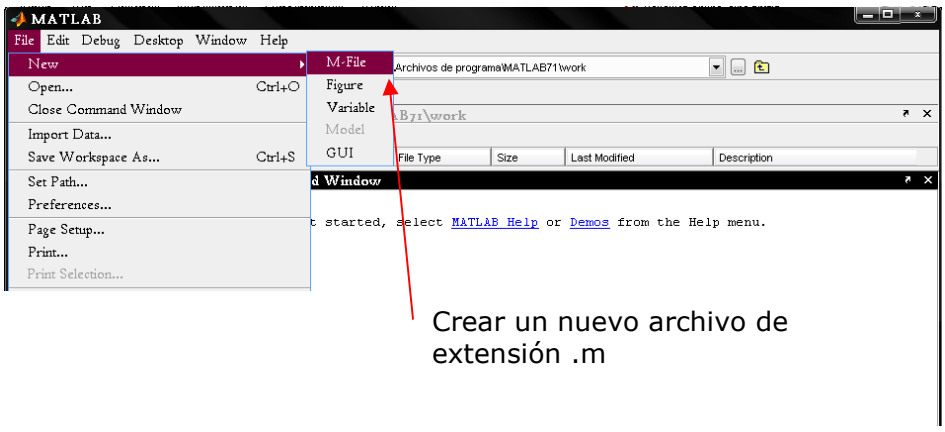
PROGRAMAS EN MATLAB

Hasta ahora se usó la ventana de comandos de MATLAB para realizar cálculos y gráficos en forma interactiva. Sabemos que también es posible crear programas, en el caso de nuestra asignatura programaremos métodos numéricos que puedan ser ejecutados varias veces e instanciados según sea el caso particular.

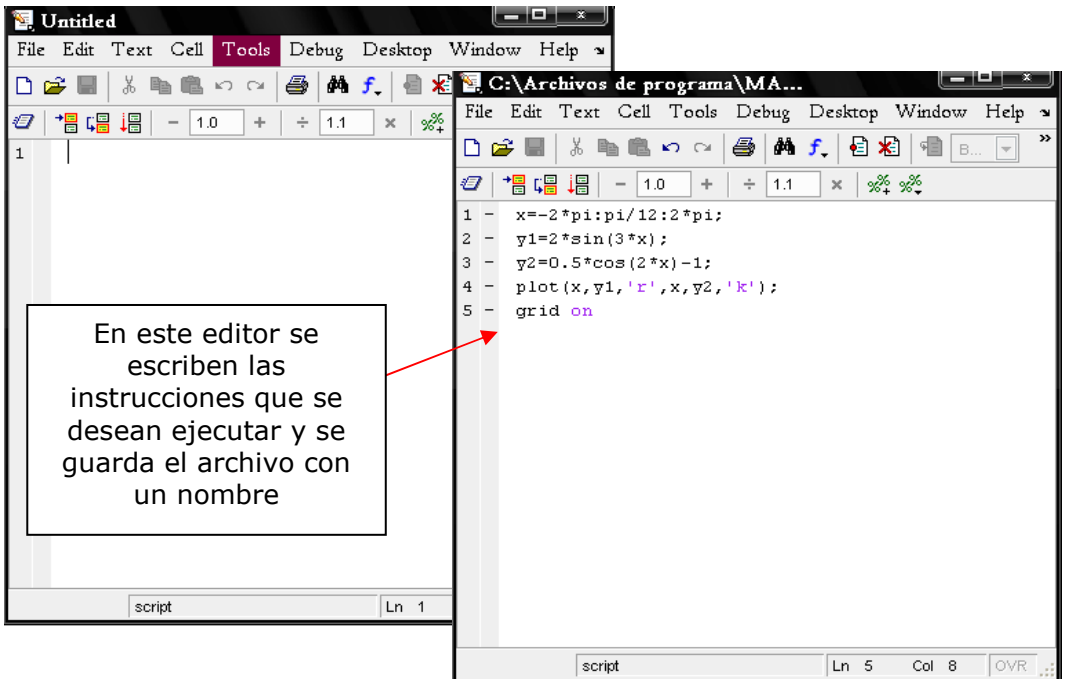
Para escribir un programa en MATLAB habrá que crear un archivo que tenga extensión .m (mediante el menú **nuevo → archivo**).

La manera más sencilla es crear un archivo de extensión m que se guarda por defecto en la carpeta work de MATLAB. Una vez escrito el programa en este archivo se invoca desde el entorno interactivo usando simplemente el nombre del mismo.

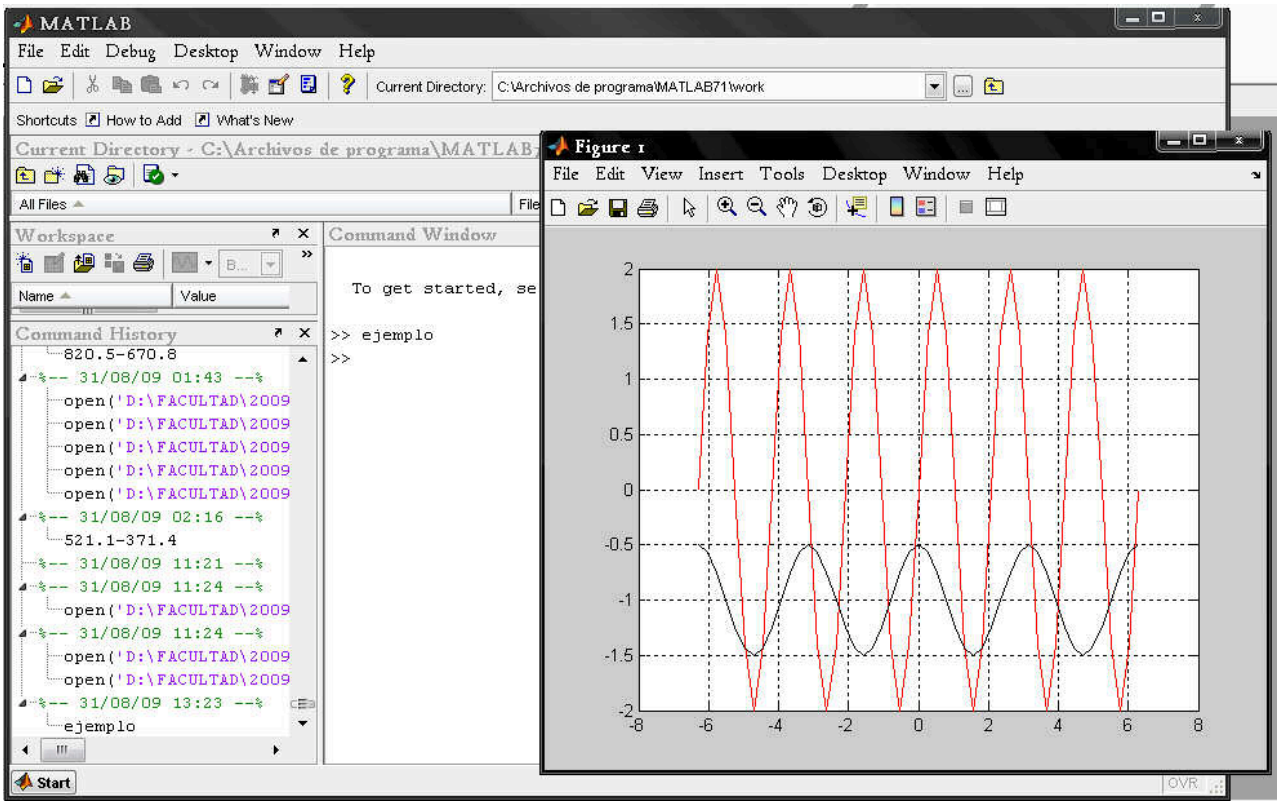
Veamos un ejemplo:



Al iniciar Matlab→ File → New → M-File y se abre un editor como el siguiente:



Para poder usar este archivo, desde la ventana de comandos se escribe el nombre del archivo y se ejecuta.



Un programa escrito en MATLAB admite la mayoría de las estructuras de programación, su uso y sintaxis son bastantes estándares.

1. Estructura IF- ELSE

   Analicemos la siguiente porción de código:

```
if x>=0
        val_abs=x;
else
        val_abs=-x;
end
```

   En este caso se evalúa la condición x>=0, si esta comparación es verdadera entonces se asigna a la variable val_abs el valor de x. Si por el contrario la condición lógica es falsa entonces a la variable val_abs se le asigna el valor –x.

   En general la estructura If-Else posee una o más condiciones lógicas cuyos resultados determinan por que rama se ejecutará el código. Esta claro que siempre se ejecutará una rama de la condición (ya sea por verdadero o falso) pero nunca ambas.

2. Estructura FOR

   Veamos un ejemplo:

```
A=[1 2 3:4 5 6;7 8 9]; % Crea una matriz A de 3x3
S=0; %Inicializa el valores de S en cero
n=3; % orden de la matriz
for k=1:n
        S=S+A(k,k); % suma el elemento A(k,k) de la diag. Ppl a lo que
                    %contenía la variable S
        disp(S);    %muestra por pantalla el valor de S
end
```

   Esta porción de código permite conseguir la suma de los elementos de la diagonal principal de la matriz A. Para cada valor de k hasta el orden de la matriz n (for k=1:n) se repite los cálculos S=S+A(k,k); y disp(S);

   Veamos una tabla que muestra como cambian los valores de la variable S en cada instante en la ejecución del código:

ESTADO	S
Inicial	0
Inicia FOR	
k=1	S+A(1,1) 0+1 1
k=2	S+A(2,2) 1+5 6
k=3	S+A(3,3) 6+9 15
Fin FOR	

3. Estructura WHILE

```
S=0; % inicializo S
n=input('Ingrese un número entero'); %pide que el usuario ingrese un num
                                      %por pantalla
while n~=0
% Mientras los números ingresados sean distintos a cero se ejecutan las
%acciones dentro del lazo
        S=S+n; % suma el número ingresado
        n=input('Ingrese un número entero'); % pide ingresar un nuevo
                                              % número
end
```

En este ejemplo se realiza la suma de todos los números que ingrese el usuario mientras éstos sean diferentes a cero. Si se ingresa un cero, el programa termina. La estructura mientras funciona ejecutando repetitivamente una o más instrucciones mientras la evaluación de la condición lógica (n~=0) sea verdadera. En general la estructura while repetirá un bloque de instrucciones dependiendo de si la condición lógica es verdadera.

Existen otras estructuras de control pero las anteriores son las más básicas y son suficientes para los alcances de este curso.
En la siguiente tabla se muestran las formas genéricas de las tres estructuras:

ESTRUCTURA IF...	ESTRUCTURA FOR...	ESTRUCTURA WHILE...
if EXP acción1; **else** acción2; **end**	**for** i=2:n acción1; acción2; ... **end**	**while** EXP acción1; acción2; ... **end**

Obs: EXP representa una expresión lógica verdadera. Por ejemplo para dos variable x, y de tipo numérico

Expresiones simples pueden ser:

x<=y
x>=y
x==y
x~=y

Expresiones complejas pueden ser:

y<10 & x<100 (y es menor que 10 y x es menor que 100)
y<10 | x<100 (y es menor que 10 ó x es menor que 100)