

# TEORÍA DE CONTROL

---

Introducción a SCILAB



## Características Básicas

**SCILAB** es un paquete de software libre de código abierto para computación científica, orientado al cálculo numérico, a las operaciones matriciales y especialmente a las aplicaciones científicas y de ingeniería.

Puede ser utilizado como simple calculadora matricial, pero su interés principal radica en los cientos de funciones tanto de propósito general como especializadas que posee así como en sus posibilidades para la visualización gráfica.

**SCILAB** posee además un lenguaje de programación propio, muy próximo a los habituales en cálculo numérico que permite al usuario escribir sus propios scripts (conjunto de comandos escritos en un fichero que se pueden ejecutar con una única orden) para resolver un problema concreto y también escribir nuevas funciones con, por ejemplo, sus propios algoritmos.

También cuenta con un entorno de simulación en diagrama en bloques (similar a Simulink de MatLab) llamado Xcos. En Xcos es posible simular sistemas de control.



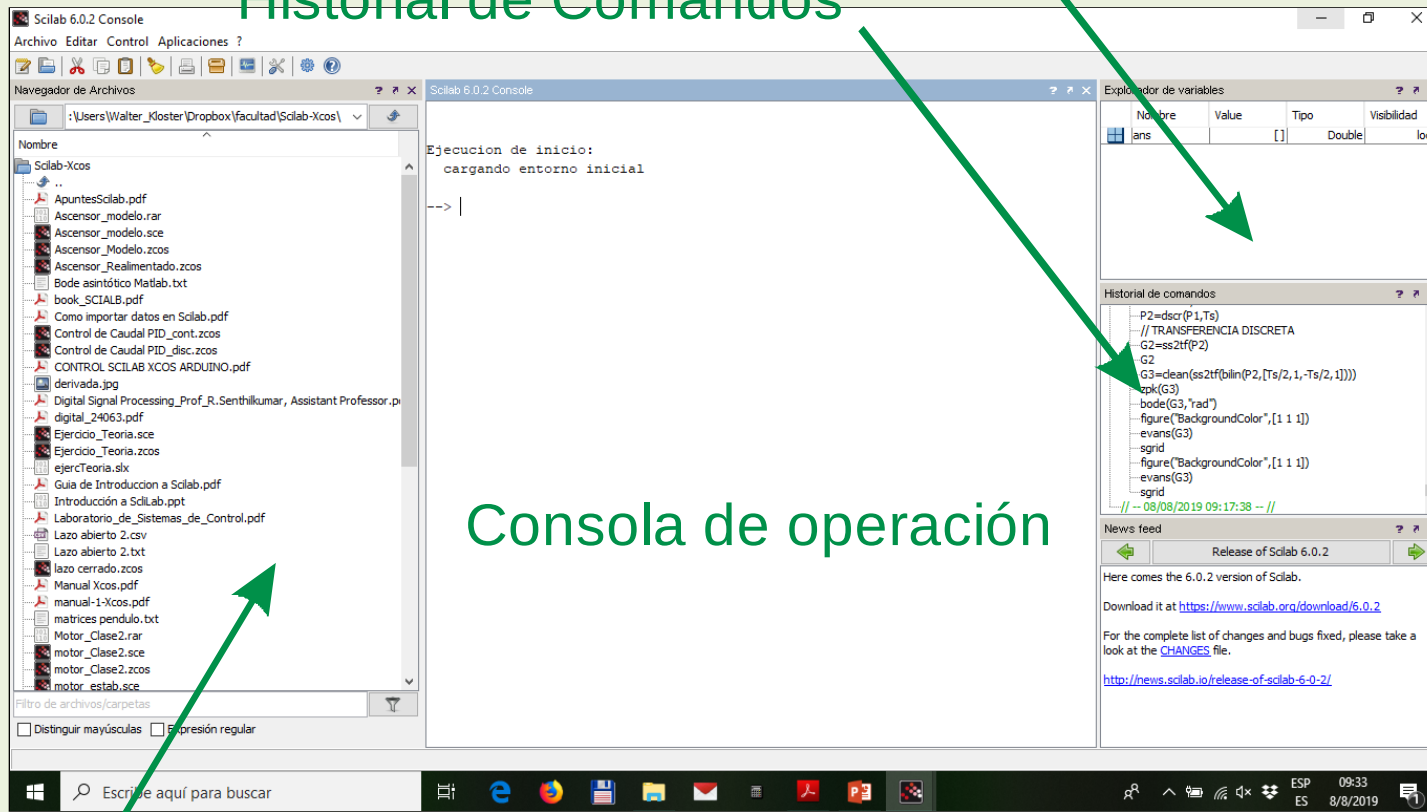
# Introducción a SCILAB

## Entorno de trabajo

El programa tiene tres aplicaciones principales.

Scilab Console

Explorador de Variables  
Historial de Comandos



Consola de operación

Navegador de Archivos

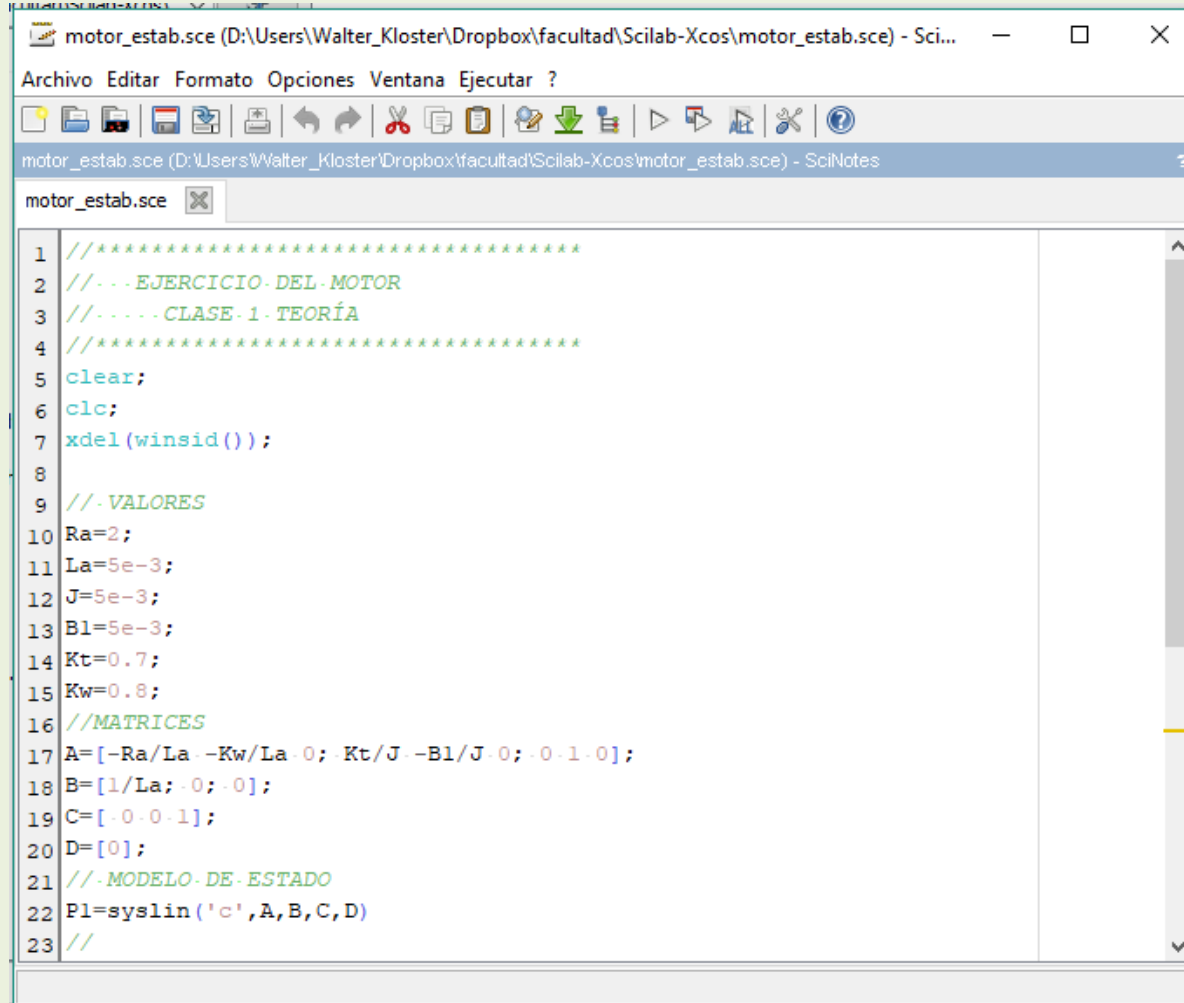
Teoría de Control



# Introducción a SCILAB

## Entorno de trabajo

### SciNotes



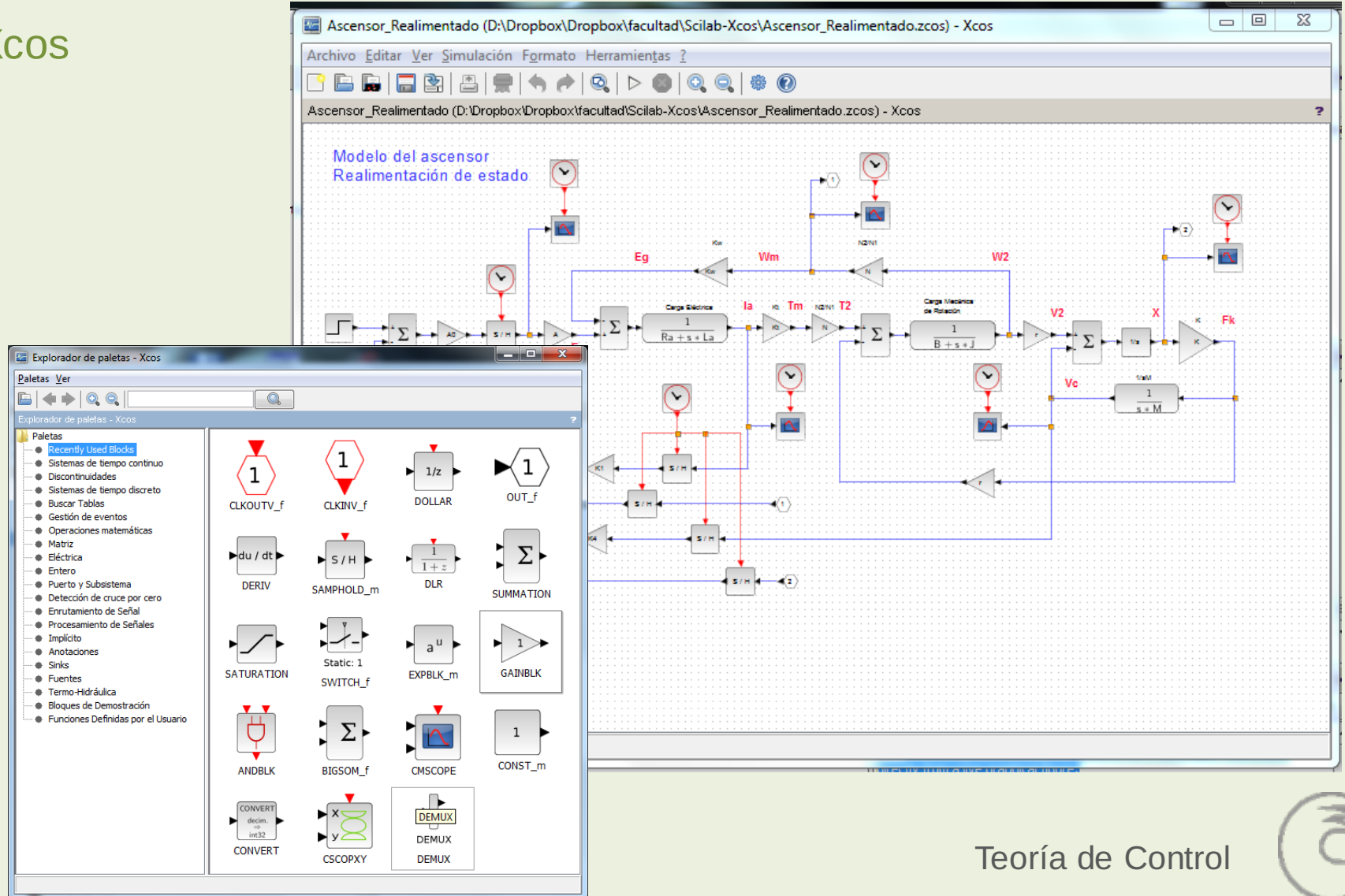
The screenshot shows the SciNotes application window titled "motor\_estab.sce (D:\Users\Walter\_Kloster\Dropbox\facultad\Scilab-Xcos\motor\_estab.sce) - Sci...". The window contains a SCILAB script with the following content:

```
1 //*****
2 //... EJERCICIO DEL MOTOR
3 //... CLASE 1 TEORÍA
4 //*****
5 clear;
6 clc;
7 xdel(winsid());
8
9 // VALORES
10 Ra=2;
11 La=5e-3;
12 J=5e-3;
13 Bl=5e-3;
14 Kt=0.7;
15 Kw=0.8;
16 // MATRICES
17 A=[-Ra/La -Kw/La 0; -Kt/J -Bl/J 0; 0 1 0];
18 B=[1/La; 0; 0];
19 C=[0 0 1];
20 D=[0];
21 // MODELO DE ESTADO
22 Pl=syslin('c',A,B,C,D)
23 //
```



## Entorno de trabajo

### Xcos



# Introducción a SCILAB

## Objetos y Sintaxis

En **SCILAB**, por defecto, los números son codificados como números reales en coma flotante en doble precisión. La precisión, esto es, el número de bits dedicados a representar la mantisa y el exponente, depende de cada tipo de máquina.

El objeto básico de trabajo de **SCILAB** es una matriz bidimensional cuyos elementos son números reales o complejos. Escalares y vectores son casos particulares de matrices. También se pueden manipular matrices de cadenas de caracteres, booleanas, enteras y de polinomios.

## Constantes

Algunas constantes numéricas están predefinidas. Sus nombres comienzan por el símbolo %.

%pi	Número $\pi$	%eps	Precisión de la máquina
%e	Base logaritmo natural	%inf	Infinito de la máquina
%i	Unidad imaginaria	%nan	Resultado de op. inválida



# Introducción a SCILAB

## Objetos y Sintaxis

El lenguaje de **SCILAB** es interpretado, es decir, las instrucciones se traducen a lenguaje maquina una a una y se ejecutan antes de pasar a la siguiente. Es posible escribir varias instrucciones en la misma línea, separándolas por una coma o por punto y coma.

**SCILAB** distingue entre mayúsculas y minúsculas: %nan **NO ES LO MISMO QUE** %Nan o que %NAN

Se pueden recuperar comandos anteriores, usando las teclas de flechas arriba y abajo. Con las flechas izquierda y derecha nos podemos desplazar sobre la línea de comando y modificarlo.

## Constantes

Reales	8.01 -5.2 0.056 1.4e+4 0.32E-2 -.567d-21 8.003D-12
Complejos	1+2*%i
Booleanos	%t %f
Caracteres	'esta es una cadena de caracteres' "string"



# Introducción a SCILAB

## Operadores

Aritméticos	+ - * / ^
Comparación	== ~= (o <>) < > <= >=
Lógicos	&   ~

## Funciones

sqrt(x)	raiz cuadrada	asin(x)	arcoseno
abs (x)	módulo	acos (x)	arcocoseno
conj(z)	complejo conjugado	atan(x)	arcotangente
rea l (z)	parte real	cosh(x)	coseno hiperbólico
imag (z)	parte imaginaria	sinh(x)	seno hiperbólico
sin(x)	seno (radianes)	tanh(x)	tangente hiperbólica
cos(x)	coseno (radianes)	acosh(x)	arcocoseno hiperbólico
tan(z)	tangente (radianes)	asinh(x)	arcoseno hiperbólico
cotg(x)	cotangente (radianes)	atanh(x)	arcotangente hiperbólica





## Funciones

<code>exp(x)</code>	exponencial	<code>rat(x)</code>	aprox. racional
<code>log(x)</code>	logaritmo natural	<code>modulo(x,y)</code>	resto de dividir x por y
<code>log10(x)</code>	logaritmo decimal	<code>floor(x)</code>	n tal que $n \leq x < (n+1)$
<code>int(x)</code>	parte entera inglesa	<code>ceil(x)</code>	n tal que $(n-1) < x \leq n$

## Uso como calculadora

Se puede utilizar Scilab como simple calculadora, escribiendo expresiones aritméticas y terminando por **RETURN** (<R>). Se obtiene el resultado inmediatamente a través de la variable del sistema `ans` (answer).

Si no se desea que Scilab escriba el resultado en el terminal, debe terminarse la orden por punto y coma.

```
Scilab 6.0.2 Console
--> sqrt(34*exp(2))/(cos(23.7)+12)
ans =

    1.3058717

--> 7*exp(5/4)+3.54
ans =

    27.972401

--> exp(1+3*i)
ans =

    -2.6910786 + 0.383604i

--> |
```



# Introducción a SCILAB

## Variables

En **Scilab** las variables no son declaradas, por lo tanto, su tipo y su tamaño cambian de acuerdo con los valores que le son asignados. Las variables se crean automáticamente al asignarles un valor.

```
Scilab 6.0.2 Console
--> a=10
a =
    10.

--> b=exp(4.7/3)
b =
    4.7906527

--> C=a+b*i
C =
    10. + 4.7906527i

--> |
```

who	Lista de variables
whos	Lista de variables detallada
who_user	Lista de variables creadas por el usuario
clear	Elimina todas las variables
clear a b c	Elimina las variables a, b y c



# Introducción a SCILAB

## Formatos

En **Scilab** las variables son representadas en formato “variable”, es decir con 10 dígitos. La función **format** permite modificar la presentación.

<code>format(20)</code>	Presentación con 20 dígitos
<code>format('e')</code>	Formato científico con punto flotante
<code>format('v')</code>	Formato variable (por defecto)
<code>format('v',20)</code>	Formato variable con 20 dígitos
<code>format('e',14)</code>	Formato científico con 14 dígitos

## Comandos útiles

<code>dir</code>	Lista de archivos del directorio
<code>pwd</code>	Devuelve el nombre y path del directorio actual
<code>cd</code>	Para cambiar de directorio
<code>clc</code>	“Limpia” la ventana de comandos
<code>date()</code>	Fecha actual



# Introducción a SCILAB

## Matrices

La forma de ingresar las matrices es entre corchetes ([...]) por filas, los elementos de la fila se separan con comas y las filas se separan con punto y coma.

```
-->v1=a:h:b // crea un vector fila de números desde a hasta un número
// c <= b, tal que c+h > b, con incrementos de h
-->v2=a:b // como el anterior, con paso h=1
-->v3=v2' // matriz traspuesta (conjugada si es compleja)
-->v4=v2.' // matriz traspuesta sin conjugar
```

```
Scilab 6.0.2 Console
--> V=[1,2,3,4]
V =

 1.  2.  3.  4.

--> W=[1;2;3;4]
W =

 1.
 2.
 3.
 4.

--> X=[1,2;3,4]
X =

 1.  2.
 3.  4.

-->
```

```
Scilab 6.0.2 Console
--> V1=1:4
V1 =

 1.  2.  3.  4.

--> V2=[V1;9,8,7,6]
V2 =

 1.  2.  3.  4.
 9.  8.  7.  6.

--> V3=V2.'
V3 =

 1.  9.
 2.  8.
 3.  7.
 4.  6.

--> |
```



# Introducción a SCILAB

## Matrices

Algunas funciones para crear matrices:

diag(x)	Si x es un vector crea una matriz diagonal con x en la diagonal principal.
diag(A)	Si A es una matriz crea un vector con la diagonal principal como elementos.
zeros(m,n)	Matriz de m filas y n columnas con todos elementos ceros
ones(m,n)	Matriz de m filas y n columnas con todos elementos unos
eye (m,n)	Matriz de m filas y n columnas con diagonal principal de unos
linspace(a,b,n)	Si a y b son números reales y n un número entero, genera un vector fila con una partición regular del intervalo [a,b] de n nodos (n-1 subintervalos)
linspace(a,b)	Igual al anterior pero con n=100
logspace(a,b,n)	Vector fila con n elementos logarítmicamente espaciados desde $10^a$ hasta $10^b$
logspace(a,b)	Igual al anterior pero con n=50
rand(m,n)	Matriz de m filas y n columnas con todos elementos aleatorios.



# Introducción a SCILAB

## Operaciones con matrices

Sean: A y B matrices de elementos  $a_{ij}$  y  $b_{ij}$  y k un escalar

$A+B$ ( $A-B$ )	matriz de elementos $a_{ij} + b_{ij}$ ( $a_{ij} - b_{ij}$ ) (con dimensiones iguales)
$A*B$	producto matricial de A y B (con dimensiones adecuadas)
$A^k$	matriz A elevada a la potencia k si k entero $>0$ , $A^k = A*A*...*A$ si k entero $<0$ , $A^k = (\text{inv}(A))^{(-k)}$
$A+k$	matriz de elementos $a_{ij} + k$
$A-k$	matriz de elementos $a_{ij} - k$
$A/k = (1/k)*A$	matriz de elementos $a_{ij} / k$
$k./A$	matriz de elementos $k / a_{ij}$
$k.^A$	matriz de elementos $k ^ ( a_{ij} )$
$A.*B$	matriz de elementos $a_{ij} * b_{ij}$ (con dimensiones iguales)
$A./B$	matriz de elementos $a_{ij} / b_{ij}$ (con dimensiones iguales)
$A.^B$	matriz de elementos $a_{ij} ^ b_{ij}$ (con dimensiones iguales)



# Introducción a SCILAB

## Funciones con matrices

La mayoría de las funciones **Scilab** están hechas de forma que admiten matrices como argumentos. Esto se aplica en particular a las funciones matemáticas elementales y su utilización debe entenderse en el sentido de "**elemento a elemento**"

<code>sum(A)</code>	suma de las componentes de la matriz A
<code>sum(A,1)</code> <code>sum(A;'r')</code>	es un vector fila (row) conteniendo la suma de los elementos de cada columna de A
<code>sum(A,2)</code> <code>sum(A;'c')</code>	es un vector columna conteniendo la suma de los elementos de cada fila de A
<code>trace(A)</code>	traza de A : <code>sum(diag(A))</code>
<code>prod(A)</code>	producto de las componentes de la matriz A
<code>prod(A,1)</code> , <code>prod(A;'r')</code>	es un vector fila (row) conteniendo el producto de los elementos de cada columna de A
<code>prod(A,2)</code> , <code>prod(A;'c')</code>	es un vector columna conteniendo el producto de los elementos de cada fila de A



## Funciones con matrices

<code>max(A)</code> <code>max(A,'r')</code> , <code>max(A,'c')</code>	máximo de las componentes de la matriz A máximos de columnas y filas resp. (como antes)
<code>mean(A)</code> <code>mean(A,'r')</code> , <code>mean(A,'c')</code>	media de las componentes de la matriz A medias de columnas y filas resp. (como antes)
<code>norm(v)</code> <code>norm(v,2)</code>	norma euclídea del vector v
<code>norm(v,p)</code>	norma-p del vector v: $\sum(\text{abs}(v).^p)^{1/p}$
<code>norm(v,'inf')</code> <code>norm(v,%inf)</code>	norma infinito del vector v: $\max(\text{abs}(v))$
<code>norm(A)</code> <code>norm(A,2)</code>	máximo autovalor de la matriz A
<code>norm(A,1)</code>	norma-1 del matriz A: máximo entre las sumas de sus columnas: $\max(\text{sum}(\text{abs}(A),'r'))$
<code>size(A)</code>	devuelve, en un vector fila, las dimensiones de la matriz A
<code>length(A)</code>	devuelve un escalar con el número de elementos de la matriz A: si A es un vector, <code>length(A)</code> es su longitud; si A es una matriz <code>length(A)</code> es el producto de sus dimensiones





## Manipulación de los elementos de una matriz

$v(i)$	i-ésima componente del vector $v$
$A(i,j)$	Elemento de la fila $i$ y columna $j$ de la matriz $A$
$A(i:j,m:n)$	Matriz formada por las filas de la $i$ a la $j$ y las columnas de la $m$ a la $n$ extraída de la matriz $A$
$A(:,n)$	n-ésima columna de la matriz $A$
$A(:)$	Representa todos los elementos de $A$ en un vector columna
$A(i,j)=k$	Reemplaza el valor de $a_{ij}$ por $k$
$A(i:j,m:n)=k$	La submatriz $A(i:j,m:n)$ se llena con $k$
$A(:,n)=[ ]$	El símbolo $[ ]$ representa una "matriz vacía". Esta instrucción ELIMINA la n-ésima columna de $A$ .



## Sistemas lineales

$A \setminus b$	Calcula la solución del sistema lineal de ecuaciones $Ax=b$ . $A \setminus b$ es como $A^{-1}b$ .
$\det(A)$	Determinante de una matriz cuadrada $A$ .
$\text{rank}(A)$	Rango de la matriz $A$ .
$\text{inv}(A)$	Matriz inversa de la matriz $A$ .
$\text{pinv}(a)$	Pseudo-inversa de la matriz $A$ . Es una matriz cuadrada $X$ tal que: $A^*X^*A = A$ , $X^*A^*X = X$ , además $A^*X$ y $X^*A$ son Hermitianas .
$[X,V]=\text{spec}(A)$	$V$ : matriz cuadrada diagonal con los autovalores y $X$ : matriz cuadrada inversible cuya $i$ -ésima columna es el autovector asociado al $i$ -ésimo autovalor.



## Polinomios

<code>poly(A,"x")</code>	Si A es una matriz cuadrada, es el polinomio característico de la matriz A, con variable simbólica x
<code>A=companion(p)</code>	Genera una matriz cuadrada llamada de Frobenius cuyo polinomio característico es p
<code>poly(v,"x","roots")</code>	Genera un polinomio con variable simbólica x cuyas raíces son los elementos del vector v
<code>poly(v,"s","coeff")</code>	Genera un polinomio con variable simbólica s cuyos coeficientes son los elementos del vector v (en potencias ascendentes)
<code>t=poly(0,"t")</code> <code>p=1+t-<math>t^2</math>+3<math>t^3</math></code>	Permite crear una variable para luego ser utilizada en la creación de polinomios en forma algebraica.
<code>roots(p)</code>	Raíces del polinomio p.
<code>horner(P,x)</code>	Evalúa el valor del polinomio P en los puntos determinados por el vector x.
<code>coeff(P)</code>	Calcula los coeficientes del polinomio
<code>pdiv(p1,p2)</code>	Cociente de polinomios



## Scripts

Un script es un conjunto de instrucciones (de cualquier lenguaje) guardadas en un fichero (usualmente de texto) que son ejecutadas normalmente mediante un intérprete. Son útiles para automatizar pequeñas tareas. También puede hacer las veces de un "programa principal" para ejecutar una aplicación.

Para generar un script en **Scilab** se utiliza el editor SciNotes, se puede acceder del menú o utilizando el comando `scinotes()`.

Por convenio, los scripts de Scilab tienen el sufijo **.sce** .

Para ejecutar un script se utiliza la expresión: `exec("nombre del archivo")`, este comando repite todas las instrucciones del script en la consola de operación. Si no se desea representar las instrucciones se puede usar el comando de la siguiente forma: `exec("nombre del archivo",-1)`.



## Funciones

Es posible definir nuevas funciones Scilab. La diferencia entre un script y una función es que esta última tiene una "interfase" de comunicación con el exterior mediante argumentos de entrada y de salida.

Las funciones **Scilab** responden al siguiente formato de escritura:

```
function [argumentos de salida] = nombre(argumentos de entrada)
// comentarios
//
...
instrucciones (normalmente terminadas por ; para evitar eco en pantalla)
...
endfunction
```

Las funciones se pueden definir on-line o bien escribiéndolas en un fichero (ASCII). A los ficheros que contienen funciones Scilab, por convenio, se les pone el sufijo **.sci** . Las funciones definidas on-line están disponibles de modo inmediato. Las funciones guardadas en un fichero hay cargarlas en el espacio de trabajo de una sesión de Scilab mediante el comando `exec`. Se pueden incluir varias funciones en un mismo fichero, una a continuación de otra.



# Introducción a SCILAB

## Funciones

También es posible definir funciones on-line mediante el comando deff:

```
deff('[arg_out]=nombre(arg_in)','instrucciones')
```

La ventaja de este tipo de función es que puede ser utilizadas para funciones sencillas, sin necesidad de crear un archivo para la función.

```
Scilab 6.0.2 Console

--> exec("fun1.sci")

--> function [y]=fun1(x)
--> // esta función calcula el resultado de un polinomio
--> y=x^2+2*x+1;
--> endfunction

--> fun1(1)
ans =

 4.

--> deff('[y]=fun2(x)', 'y=3*x^2-5*x+5')

--> fun2(1)
ans =

 3.

-->
```

```
fun1.sci (D:\Dropbox\Dropbox\facultad\Scilab-Xcos\fun1.sci) - SciNotes
Archivo Editar Formato Opciones Ventana Ejecutar ?
fun1.sci (D:\Dropbox\Dropbox\facultad\Scilab-Xcos\fun1.sci) - SciNotes
fun1.sci
1 function [y]=fun1(x)
2 > // esta función calcula el resultado de un polinomio
3 > y=x^2+2*x+1;
4 endfunction
5
```



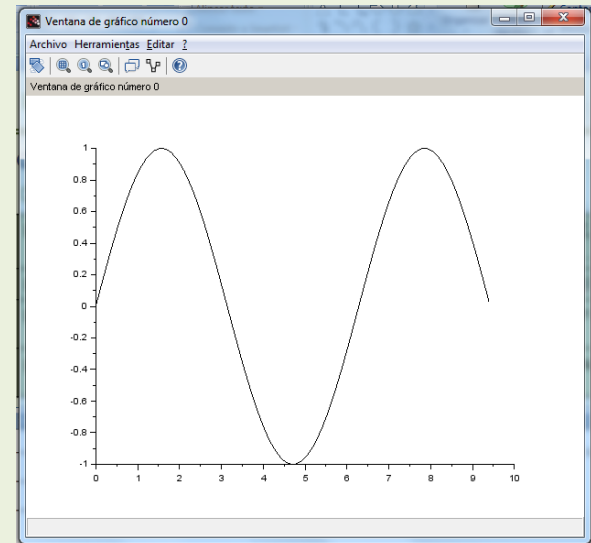
# Introducción a SCILAB

## Gráficos

### Curvas Planas

<code>plot2d(x,y)</code>	Dados dos vectores $x=[x_1,x_2,\dots,x_n]$ e $y=[y_1,y_2,\dots,y_n]$ dibuja la curva que pasa por los puntos $(x_1,y_1)\dots(x_n,y_n)$
<code>plot2d(y)</code>	dibuja la curva que pasa por los puntos $(1,y_1)\dots(n,y_n)$
<code>plot2d(x,y)</code>	siendo $x$ un vector e $y$ una matriz dibuja una curva por cada columna de $y$
<code>plot2d(x,y)</code>	siendo $x$ e $y$ matrices de las mismas dimensiones, dibuja una curva por cada par ( columna de $x$ , columna de $y$ )

```
Scilab 6.0.2 Console
--> x=[0:0.1:3*pi]'; //Genero la variable x que va desde 0 a 3 * pi y con intervalos de 0,1.
--> plot2d(x,sin(x)); //Ploteo de la función.
--> |
```



## Gráficos

Observaciones:

- Por defecto, gráficas sucesivas se superponen. Para evitarlo, hay que borrar la gráfica anterior antes de dibujar de nuevo. Ello puede hacerse ó bien cerrando la ventana gráfica ó bien borrando su contenido desde la barra de menús ( Edit --> Limpiar Figura) o mediante un comando ( delete()).
- Cuando se borra el contenido de la ventana gráfica, pero no se cierra, se conservan sus características. Si, por ejemplo, se ha modificado la carta de colores de esa ventana, se seguirá conservando la carta al borrarla, pero no al cerrarla.
- Cuando plot2d dibuja varias curvas, les asigna distintos colores. El orden de los colores asignados viene determinado por la carta de colores activa. Por defecto, es la siguiente:

<b>1</b>	<b>7</b>				
<b>2</b>					
<b>3</b>					
<b>4</b>					
<b>5</b>					
<b>6</b>					





## Gráficos

<code>subplot(m,n,p)</code>	divide la ventana gráfica en una matriz $m \times n$ y utiliza la caja $p$ para el siguiente dibujo que se indique.
<code>title("Mi Título")</code>	Título del gráfico
<code>xlabel("Eje x")</code> <code>ylabel("Eje y")</code> <code>zlabel("Eje z")</code>	Leyenda de los ejes $x$ , $y$ y $z$
<code>xtitle( 't', 'x', 'y', 'z' )</code>	siendo $t$ =título del grafico, $x$ =leyenda del eje $x$ , $y$ =leyenda del eje $y$ , $z$ =leyenda del eje $z$
<code>xgrid(c)</code>	dibuja la grilla del color $c$
<code>legend("str")</code>	Agrega una leyenda correspondiente al color de la gráfica



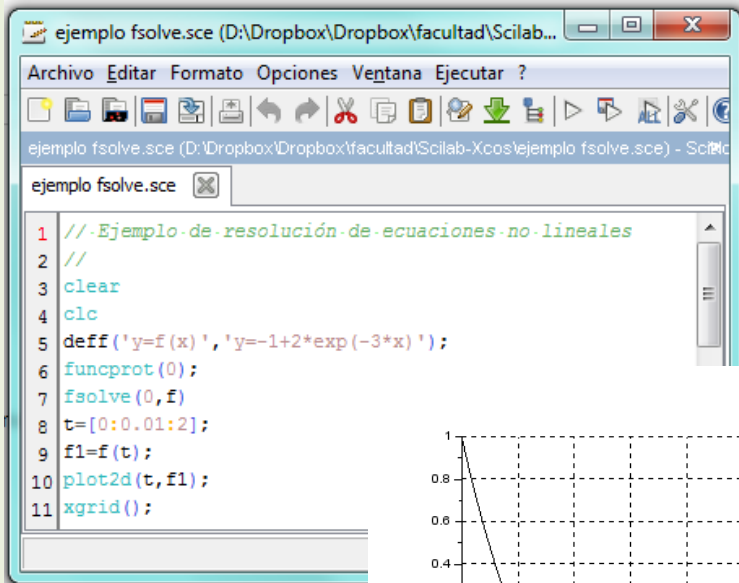
# Introducción a SCILAB

## Cálculo

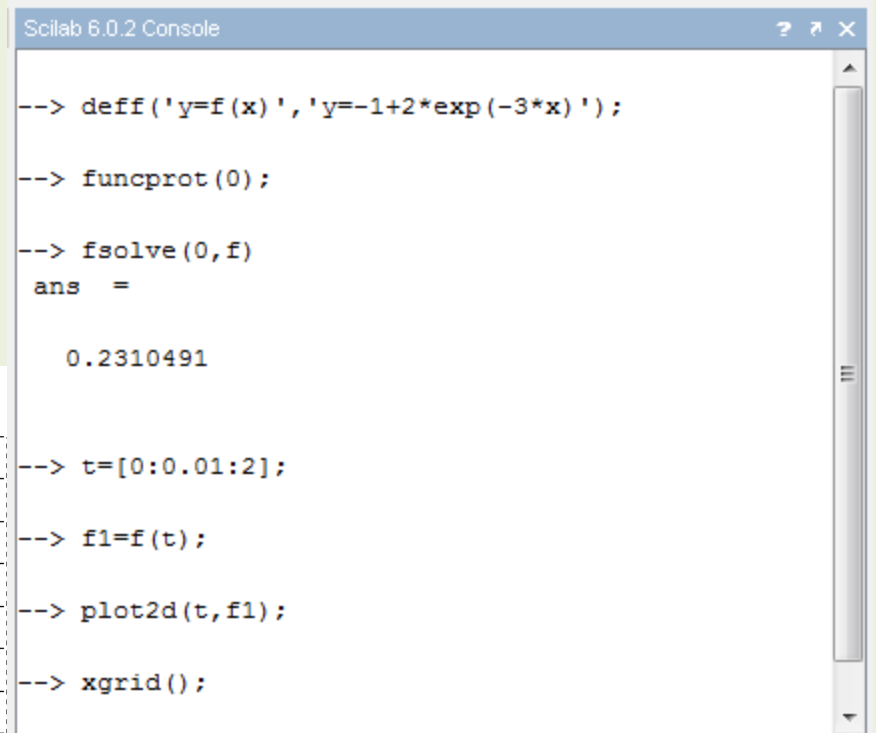
### Resolución de ecuaciones no lineales

`[x ,v]=fsolve(x0,func)`

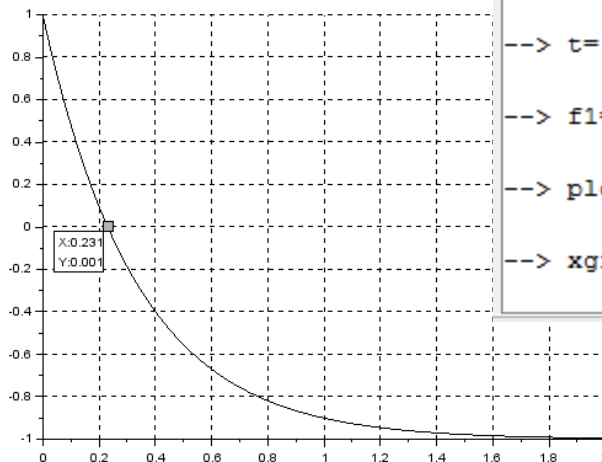
x: solución de la ecuación      v: valor de la función en x  
x0: valor inicial para el cálculo.  
func: función a resolver



```
1 // -Ejemplo de resolución de ecuaciones no lineales
2 //
3 clear
4 clc
5 deff('y=f(x)', 'y=-1+2*exp(-3*x)');
6 funcprot(0);
7 fsolve(0, f);
8 t=[0:0.01:2];
9 f1=f(t);
10 plot2d(t, f1);
11 xgrid();
```



```
Scilab 6.0.2 Console
--> deff('y=f(x)', 'y=-1+2*exp(-3*x)');
--> funcprot(0);
--> fsolve(0, f)
ans =
    0.2310491
--> t=[0:0.01:2];
--> f1=f(t);
--> plot2d(t, f1);
--> xgrid();
```

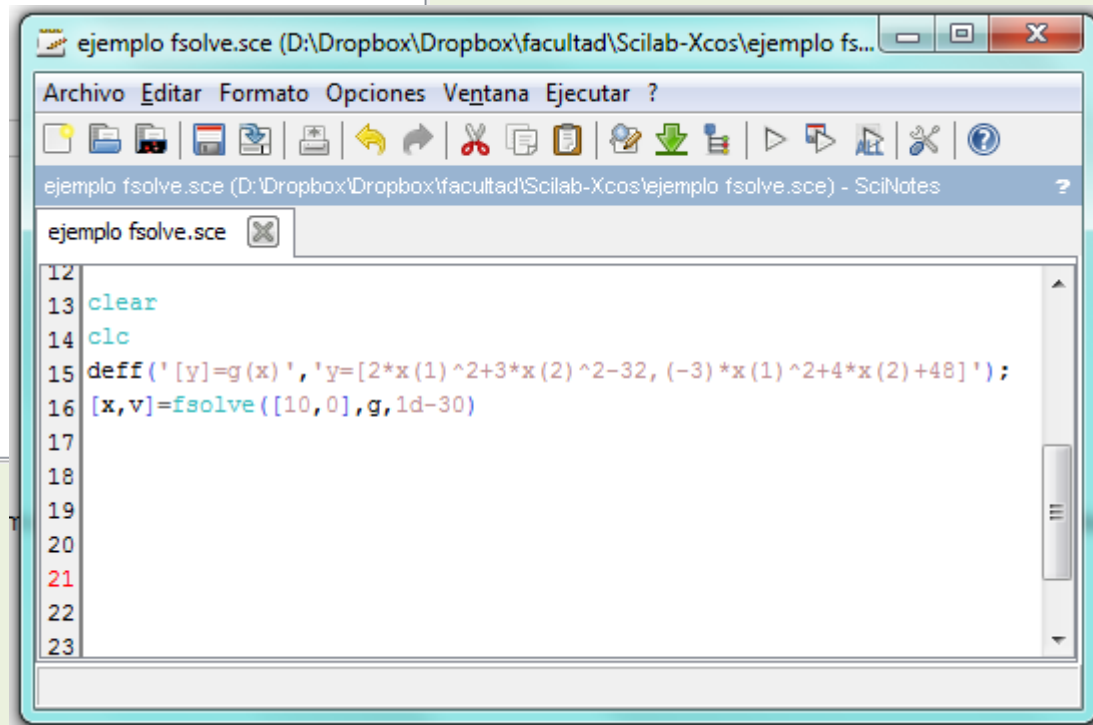


# Introducción a SCILAB

## Cálculo

### Resolución de sistemas de ecuaciones no lineales

```
Scilab 6.0.2 Console
--> deff(' [y]=g(x) ', 'y=[2*x(1)^2+3*x(2)^2-32, (-3)*x(1)^2+4*x(2)+48] ');
--> [x,v]=fsolve([10,0],g,1d-30)
v =
  0.    0.
x =
  4.  -1.310D-16
--> |
```



The screenshot shows a Scilab editor window titled 'ejemplo fsolve.sce'. The code in the editor is as follows:

```
12
13 clear
14 clc
15 deff(' [y]=g(x) ', 'y=[2*x(1)^2+3*x(2)^2-32, (-3)*x(1)^2+4*x(2)+48] ');
16 [x,v]=fsolve([10,0],g,1d-30)
17
18
19
20
21
22
23
```



# Introducción a SCILAB

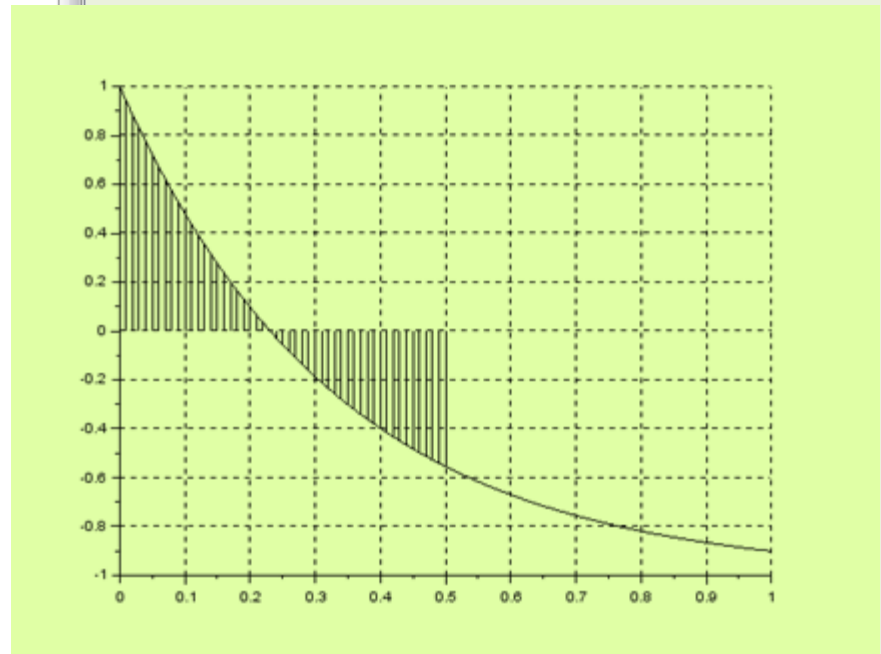
## Cálculo

### Integrales definidas

`intg(a,b,func)`

a: límite inferior de la integral    b: límite superior de la integral  
func: función a integrar

```
Scilab 6.0.2 Console
--> deff('y=f(x)', 'y=-1+2*exp(-3*x)');
--> funcprot(0);
--> intg(0,0.5,f)
ans =
    0.0179132
--> t=[0:0.01:1];
--> f1=f(t);
--> plot2d(t,f1);
--> xgrid();
--> t=[0:0.01:0.5];
--> f1=f(t);
--> plot2d3(t,f1);
--> |
```



# Introducción a SCILAB

## Cálculo

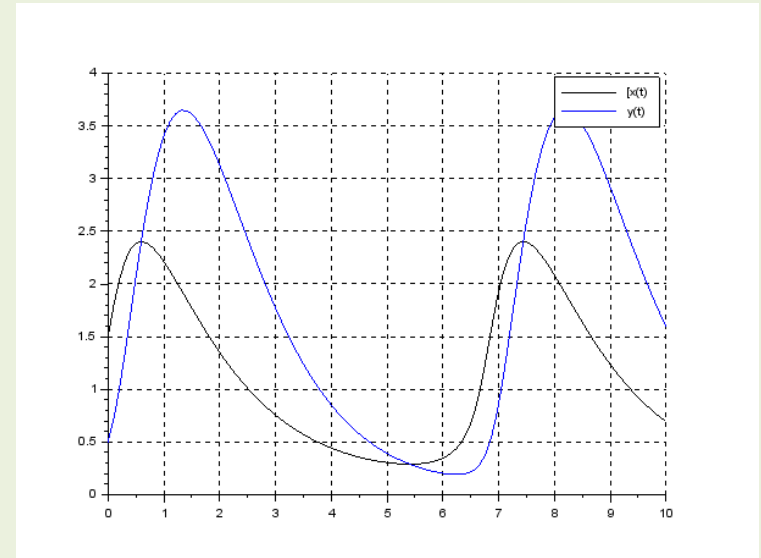
### Ecuaciones diferenciales

$y = \text{ode}(y_0, t_0, t, f)$

- $y_0$  es el valor de la condición inicial
- $t_0$  es el punto en que se impone la condición inicial.
- $t$  es el vector que contiene los valores de  $t$  para los cuales se quiere calcular la solución
- $f$  es el nombre de la función que calcula
- $y$  es un vector conteniendo los valores de la solución

$$\begin{cases} \dot{x} = \frac{x^2}{y} - x & x(0) = 1.5 \\ \dot{y} = x^2 - y & y(0) = 0.5 \end{cases}$$

```
function [dydx]=fty(t,y), dydx=[(y(1)^2/y(2)-y(1));y(1)^2-y(2)],endfunction
t0=0;
y0=[1.5;0.5];
tf=10;
t=linspace(t0,tf);
y=ode(y0,t0,t,fty);
delete();
plot2d(t',y')
xgrid();
legend("x(t)","y(t)")
```



# Introducción a SCILAB

## Programación

### Condicionales

Permite la ejecución de un conjunto de instrucciones si se cumple una condición.

<pre>if condición [then]   instrucciones; end</pre>	<pre>if condición [then]   instrucciones; else   instrucciones; end</pre>	<pre>if condición 1[then]   instrucciones; elseif condición 2   instrucciones; else   instrucciones; end</pre>
---	---	--

<pre>select variable   case valor1 [then]     Instrucciones;   case valor2 [then]     Instrucciones;   .... else   Instrucciones; end</pre>	<p>Esta sentencia compara el valor de la variable con cada valor de los case, si coincide el valor ejecuta las instrucciones, si no coincide con ninguna opción ejecuta las instrucciones de else.</p>
---	--



# Introducción a SCILAB

## Programación

### Bucles

Son estructuras que repiten una serie de instrucciones según una condición lógica o un rango de valores.

For: Repite las instrucciones hasta llegar al fin del rango de valores	While: Repite las instrucciones mientras una condición sea verdadera
<pre>for índice=rango   instrucciones; end</pre>	<pre>while condición   instrucciones; end</pre>

