Repaso. Scilab

1. Matrices y vectores en scilab
2. Funciones length() y size()
3. Acceso a elementos de los vectores y matrices
4. Operador "dos puntos"
5. Condicionales. Palabra clave "if"
6. Bucles "for" y "while"
7. Uso los comandos "break" y "continue"
8. Los bucles anidados
9. Funciones.
10. Funciones disp() y printf()
11. Gráficas

1. Matrices y vectores en scilab

Comando **A=[1, 2, 3]** o **A=[1 2 3]** crea un vector renglón.

Un vector columna puede ser definida con el comando **B=[1; 2; 3; 4]** o **B=[1 2 3 4]'**. (apostrofo después de corchete significa "matriz transpuesta")

Crear una matriz 3x4

C=[1, 2, 3, 4; 5, 6, 7, 8; -9, 10, -11, 12]

```
C  =    1.    2.    3.    4.
        5.    6.    7.    8.
      - 9.   10.  - 11.   12.
```

Some basic matrices can be generated with a single command:

zeros(m,n) - matriz de zeros, m renglones, n columnas;

ones(m,n) - matriz de unos, m renglones, n columnas;

eye(m) – matriz de identidad (having 1 in the main diagonal and 0 elsewhere)

rand() - random elements (follows either normal or uniform distribution)

2. Funciones length() y size()

muestran longitud del vector y tamaño de matriz

Para B=[1 2 3 4]' y C=[1, 2, 3, 4; 5, 6, 7, 8; -9, 10, -11, 12]

length(B) nos da 4,
length(C) nos da 12
size(B) nos da [1, 4] – el tamaño del vector es 1 por 4
size(C) nos da [3, 4] – el tamaño de la matriz es 3 por 4

3. Acceso a elementos de los vectores y matrices

```
C  =    1.    2.     3.     4.
        5.    6.     7.     8.
      - 9.   10.   - 11.   12.
```

A elemento de matriz

```
C(2,2)
 ans  =    6.
```

A segundo renglón

```
C(2,:)
 ans  = 5.    6.    7.    8.
```

A tercer columna

```
C(:,3)
 ans  =
    3.
    7.
```

- 11.

A bloque

C(2:3,3:4)

ans =

7. 8.

- 11. 12.

4. Operador "dos puntos"

"**a:b**" Genera elementos entre **a** y **b**

```
1:10
ans =1. 2. 3. 4. 5. 6. 7. 8. 9. 10.
```

"**a:b:c**" Genera elementos entre **a** y **c** con el paso **b**

```
10:-0.1: 9
 ans  = 10.    9.9    9.8    9.7    9.6    9.5    9.4    9.3    9.2    9.1    9.
```

5. Condicionales. Palabra clave "if"

keyword for conditional execution

Calling Sequence

```
if expr1 then
   statements
elseif expri then
   statements
....
else
   statements
end
```

Ejemplos de "if"

```
// programa que verifica edades

clc
clear

x=input('Introduce su edad')

if x<18
   disp('Menor de edad')
else
   disp('Mayor de edad')
end
```

Ejemplos de "if"

```
clc
clear
//programa que dice su status basando a una calificación
disp('Hola')
disp('')
calif=input('Introduce su calificación >');
if calif<4
   disp('Nos vemos en el extra')
elseif calif>=4 & calif<7
   disp('Nos vemos en el remedial')
elseif calif>=7 & calif<10
   disp('Te salvaste, estas aprobado')
elseif calif==10
   disp('Eres un genio')
else
   disp('Favor de teclear un numero entre 0 y 10')
end
```

6. Bucles.

Los bucles (loops) se usan cuando necesita repetir un conjunto de instrucciones muchas veces. Scilab soporta dos tipos diferentes de bucles: el bucle "**for**" y el bucle "**while**". Los bucles "**for**" son la opción más sencilla cuando usted sabe cuántas veces necesita repetir el bucle.

Palabra clave para los bucles "for"

Used to define loops. Its syntax is:

```
for variable=expression
   instruction
       ...
   instruction
end
```

Ejemplo "for"

```
clc
clear

x=zeros(1,10)

for i=1:10
  x(i)=i
end
disp(x)

////////////////////////////////
```

7. Uso los comandos "break" y "continue"

```
x=zeros(1,10)
for i=1:10
  if(i>5)
    break //salto a fin del ciclo
  end
  x(i)=i
end
disp(x)
////////////////////////////////////
x=zeros(1,10)
for i=1:10
  if(i==5)
    continue ////salto a principio del ciclo
  end
  x(i)=i
end
disp(x)
```

1. Que hace función **sum**()?

For an array x, y=**sum**(x) returns in the scalar y the sum of all the elements of x.

$$y=\sum_i x_i$$

2. Desarrollar un función MySum()

```
function y=MySum(x)
    n=length(x)
    y=0
    for i=1:n
        y=y+x(i)
    end
endfunction
x=[1,2,3,4,5]
disp(x)
y=MySum(x)
disp(y,"sum of element is ")
```

Que hace función **abs**()?

abs(x) is the absolute value of the elements of x.

Desarrollar un función MyAbs() (si valor de elemento es menos que 0, multiplicar este elemento por -1)

```
function y=MyAbs(x)
    n=length(x)
    y=zeros(x)
    for i=1:n
      if(x(i)<0)
        y(i)=-1*x(i)
      else
        y(i)=x(i)
      end
    end
endfunction


x=[1,2,3,-4,-5]
disp(x)
```

```
y=MyAbs(x)
disp(y)
```

8. Los bucles anidados

```
for i=1:2
  printf("Ciclo externo,i=%d \n",i)
  for j=1:3
    printf("Ciclo interno,i=%d, j=%d \n",i,j)
  end
end

//Resultados
Ciclo externo,i=1
Ciclo interno,i=1, j=1
Ciclo interno,i=1, j=2
Ciclo interno,i=1, j=3
Ciclo externo,i=2
Ciclo interno,i=2, j=1
Ciclo interno,i=2, j=2
Ciclo interno,i=2, j=3
```

El uso de los bucles anidados para los matrices.

```
//Tablas de multiplicar de tamaño nxn

n=10

// Solucion 1
A = zeros(n,n);
for j = 1:n
  for k=1:n
    A(j,k) = j*k;
  end
end
disp(A)
```

```
// Solucion 2
B = [];
k = 1:n;
for j = 1:n
  B(j,k) = j*k;
end
disp(B)


// Solucion 3
k = 1:n;
C = k' * k
disp(C)
```

El resultado es

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
2.	4.	6.	8.	10.	12.	14.	16.	18.	20.
3.	6.	9.	12.	15.	18.	21.	24.	27.	30.
4.	8.	12.	16.	20.	24.	28.	32.	36.	40.
5.	10.	15.	20.	25.	30.	35.	40.	45.	50.
6.	12.	18.	24.	30.	36.	42.	48.	54.	60.
7.	14.	21.	28.	35.	42.	49.	56.	63.	70.
8.	16.	24.	32.	40.	48.	56.	64.	72.	80.
9.	18.	27.	36.	45.	54.	63.	72.	81.	90.
10.	20.	30.	40.	50.	60.	70.	80.	90.	100.

Bucle "while"

Se usan cuando no saben cuanto veces quieren repetir el bucle.

Formato

while cond
    instrucciones,
[**else instrucciones**],
end

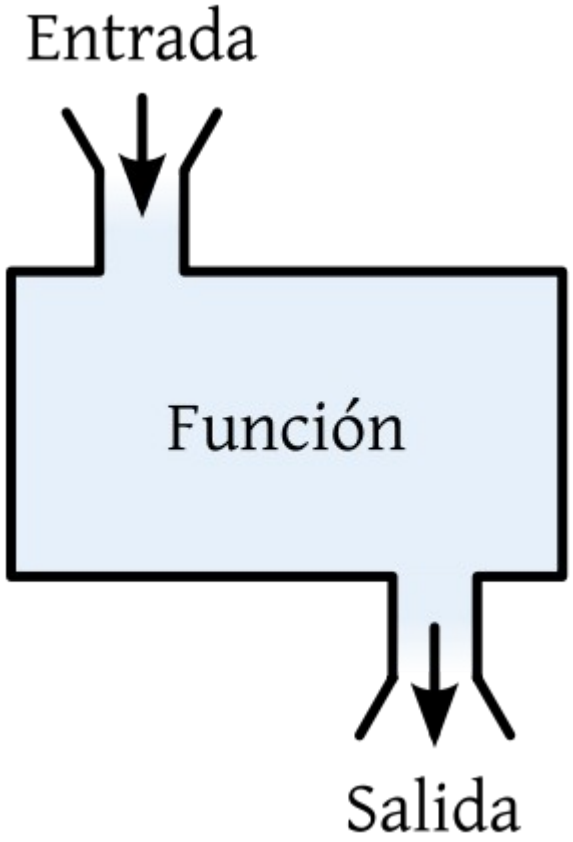
The "**while"** must be terminated by "**end**".

El bucle se repita hasta condición "**cond**" es verdad.

Ejemplo

```
i = 0
while i<5
    disp(i,"i=");
    i = i + 1;
end

e=1; a=1; k=1;
while abs(a-(a+e)) > %eps,
    e=e/2;
    k=k+1;
end
disp(e,"e=")
disp(k,"k=")
```

9. Funciones



En matemática.

Función, **aplicación** o **mapeo** es una regla que asigna a cada elemento de un primer conjunto un único elemento de un segundo conjunto (correspondencia matemática). Por ejemplo, cada número entero posee un único cuadrado, que resulta ser un número natural (incluyendo el cero):

-2->+4, -1->+1, 0->0, 1->1, 2->4...

En Ciencias de la Computación.

Función es un subprograma o subrutina que realiza una tarea específica y devuelve un valor.

Ejemplos

```
//Definición de función
function y=sqr(x)
    y=x*x
endfunction
// El uso
y=sqr(2)
disp(y,"y=")

//inline definition (see function)
function [x, y]=myfct(a, b)
x=a+b
y=a-b
endfunction

[x,y]=myfct(3,2)
//nested functions definition
```

```
function y=foo(x)
 a=sin(x)
 function y=sq(x)
   y=x^2
 endfunction
 y=sq(a)+1
endfunction

foo(%pi/3)
```

// **definition in a script file** (codigo de funcion esta guardado en el archivo asi-nh.sci en directorio SCI/modules/elementary_functions/macros)

```
exec SCI/modules/elementary_functions/macros/asinh.sci;
```

10. Funciones disp() y printf()

Para mostrar e imprimir en ventana de comandos.

Exemplos

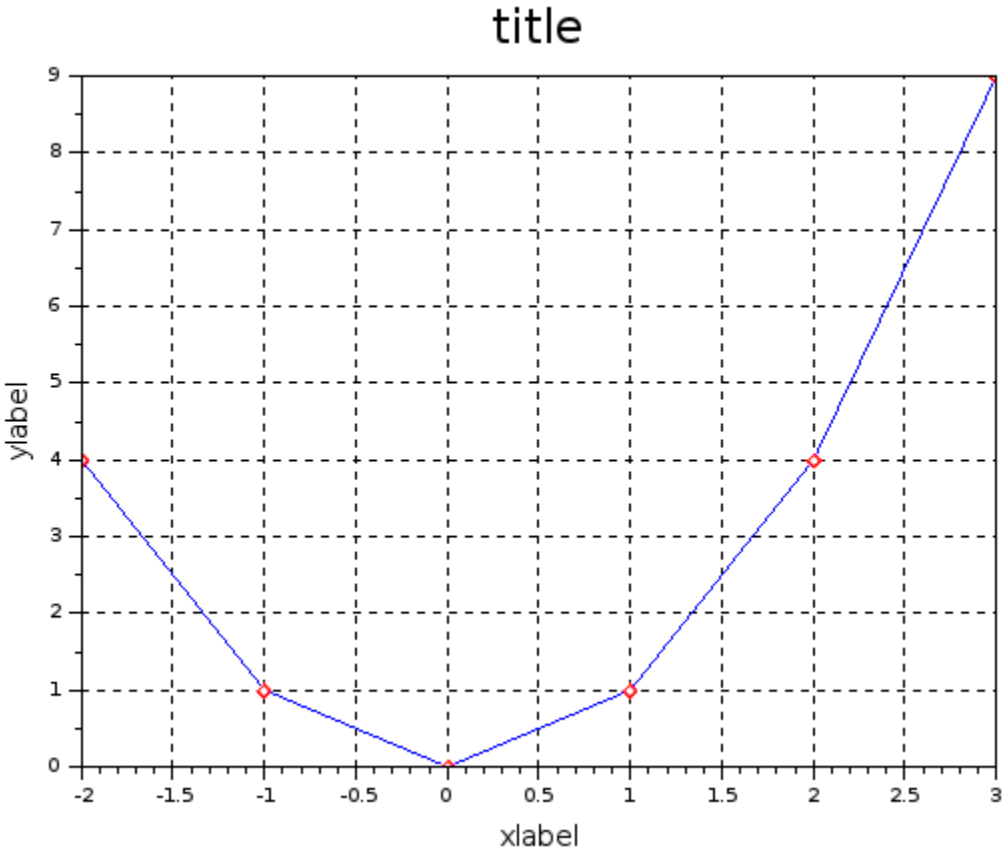
```
disp("Texto")

a=3;  disp(a,"a=")//texto y numeros

b=[1,2;3,4];  disp(b,"b=")//texto y matriz

printf("\n") // nueva linea

printf("Texto\n")

printf("a=%i\n",a)//texto y numeros
//%i,%d para enteros

printf("a=%f\n",a)//texto y numeros
```

```
//%f,%g para números reales

printf("\nMatriz b=\n")//texto
printf("%f %f\n",b)// matriz
```

11. Gráficas

El comando **plot** genera una gráfica en 2D

```
x = [-2 -1 0 1 2 3];
y = [4 1 0 1 4 9];
plot (x,y)
```



Distintos tipos de líneas para el dibujo de la gráfica:

```
plot (x,y,"rd")
```

Etiquetas

Etiqueta el eje X de la gráfica:

```
xlabel("xlabel","fontsize",3)
```

Etiqueta el eje Y de la gráfica: `ylabel("ylabel","fontsize",3)`

Título en la cabecera de la gráfica: `title("title","fontsize",5)`

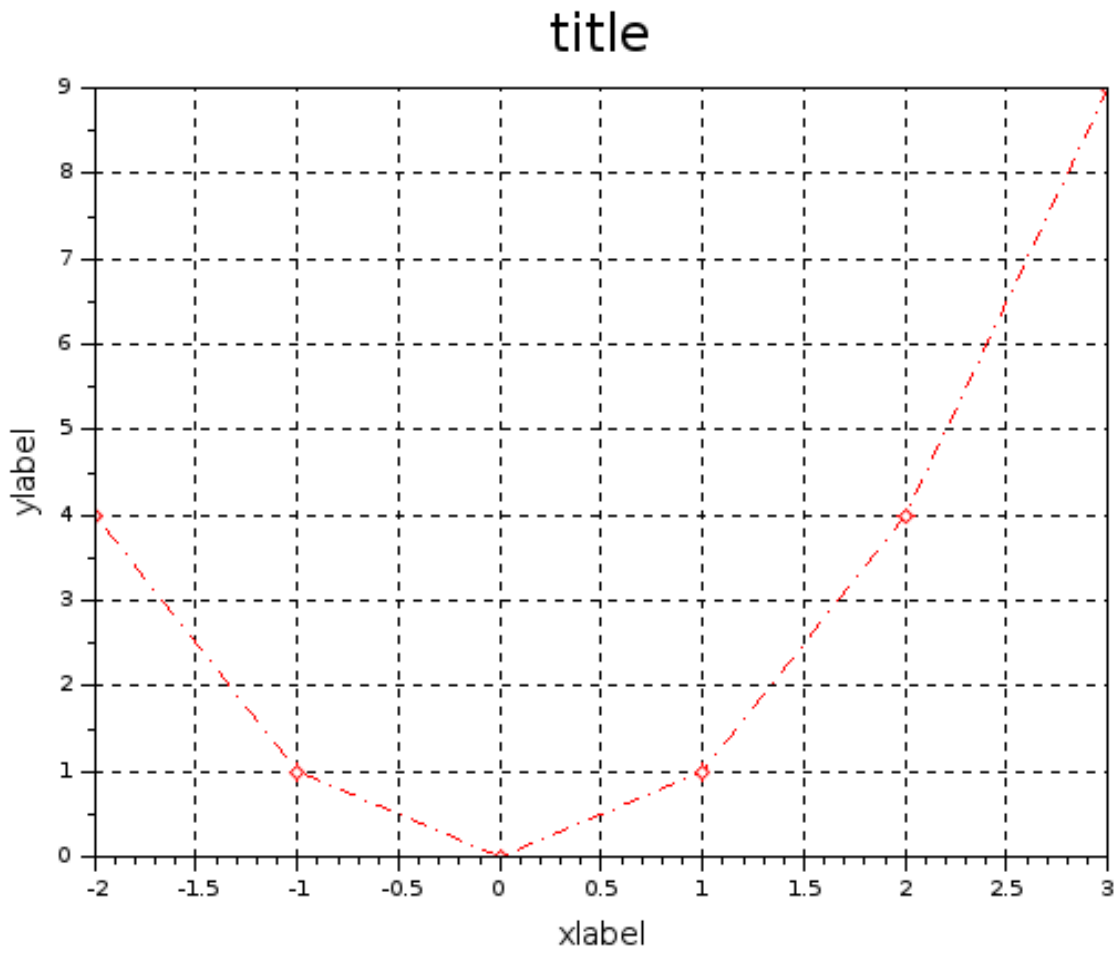
Malla: `xgrid()`

Gráficas. Sintaxis

plot(x1,y1,<LineSpec1>,x2,y2,<LineSpec2>,...)

LineSpec - para personalizar cada aspecto de linea nueva

To specify a red long dash-dot with diamond marker, the string can be "r-.diam". Por ejemplo plot (x,y,"r-.diam")



LineStyle: a string defining the line style.

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dotted line

Color: a string defining the line color.

Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black

Marker type:

Note that if you specify a marker without a line style, only the marker is drawn.

Specifier	Marker Type
+	Plus sign
o	Circle
*	Asterisk
.	Point
x	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)

Funciones para trabajo con ventanas gráficas

scf() - set the current graphic figure (window)

clf() - Clear or reset a figure.

gcf() - Return handle of current graphic window.

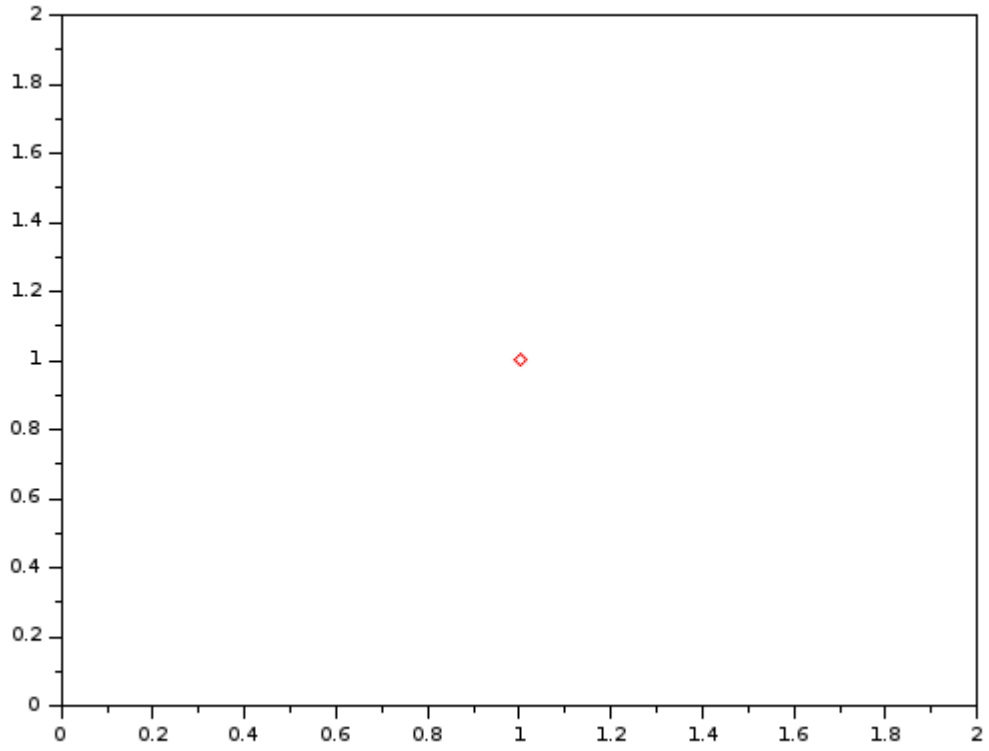
gdf() - Return handle of default figure.

set() - set a property value of a graphic entity object.

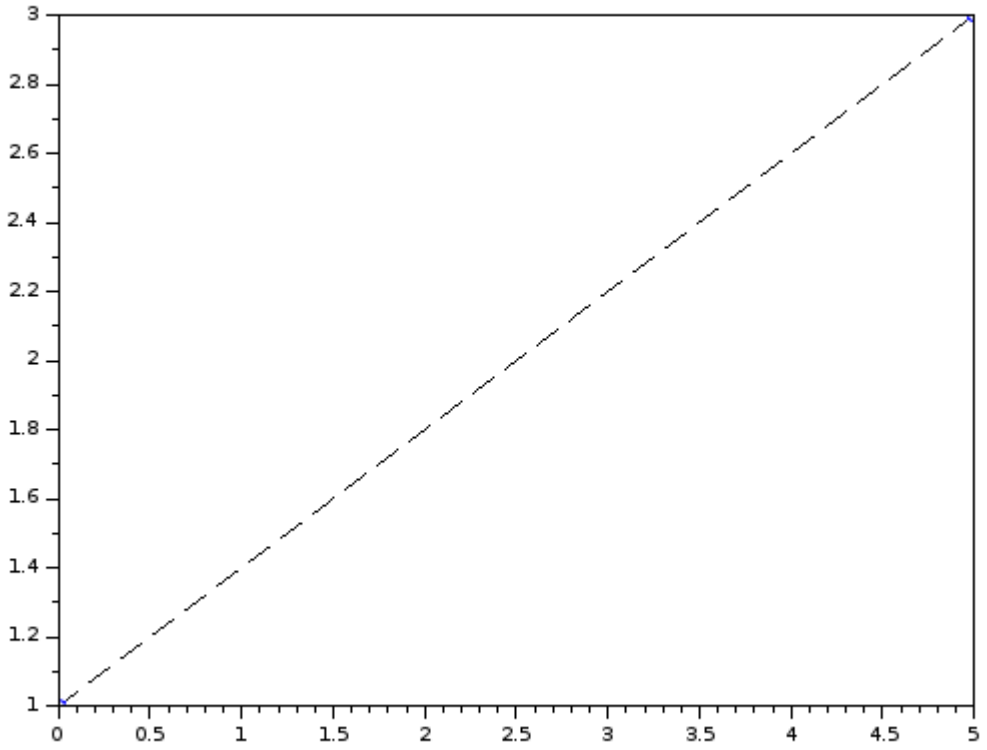
get() - Retrieve a property value from a graphics entity or an User Interface object.

Ejemplos

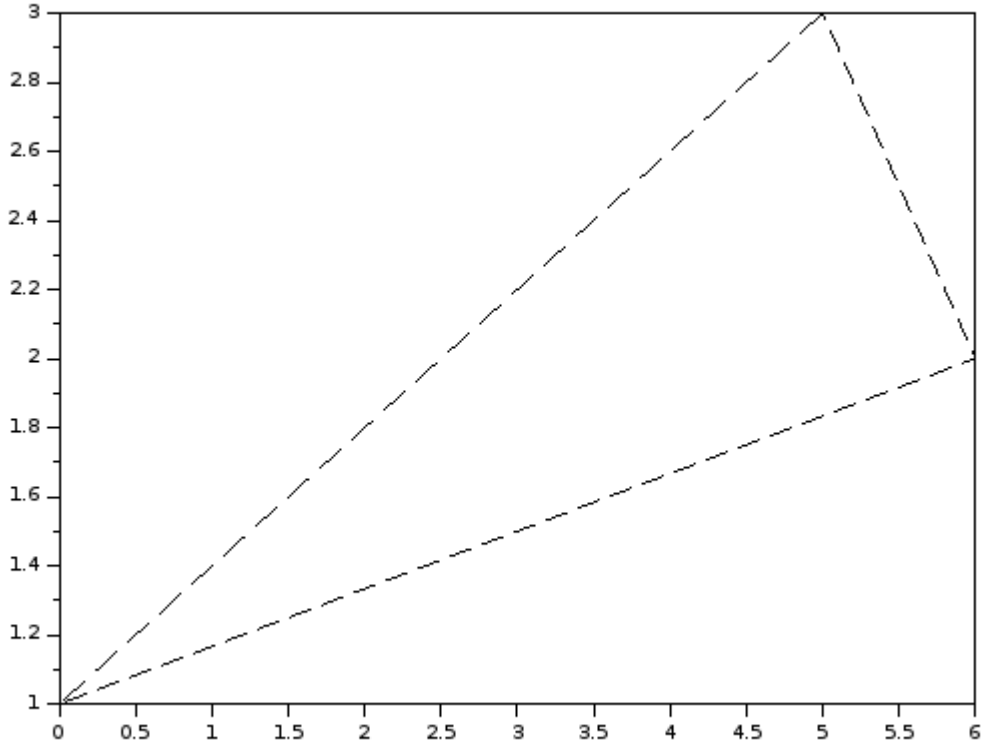
```
//dibujar un punto P(1,1)
//eso sigunifica que coordenadas
x=1
y=1
f1=scf(1)
plot(x,y,"rd")//rd es red diamonds
```



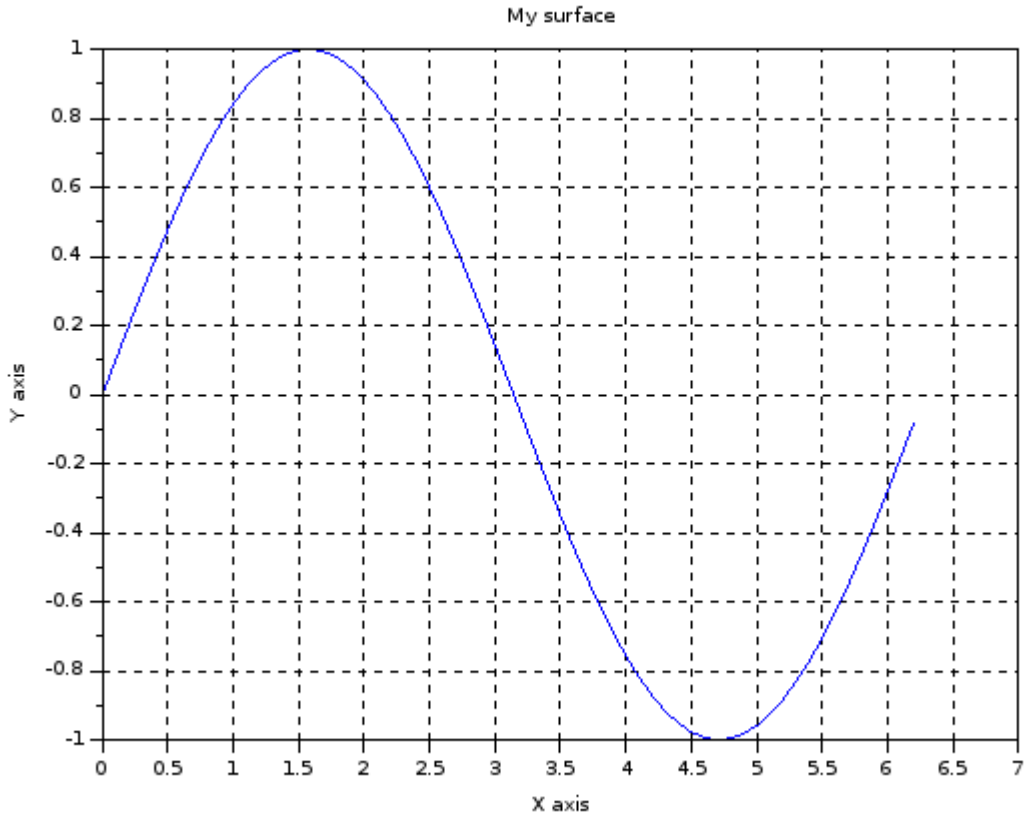
```
//dibujar una linea en puntos
P1=[5,3]
P2=[0,1]

x=[P1(1),P2(1)]
y=[P1(2),P2(2)]

f2=scf(2)
plot(x,y,"k--")
plot(x,y,"bo")
```



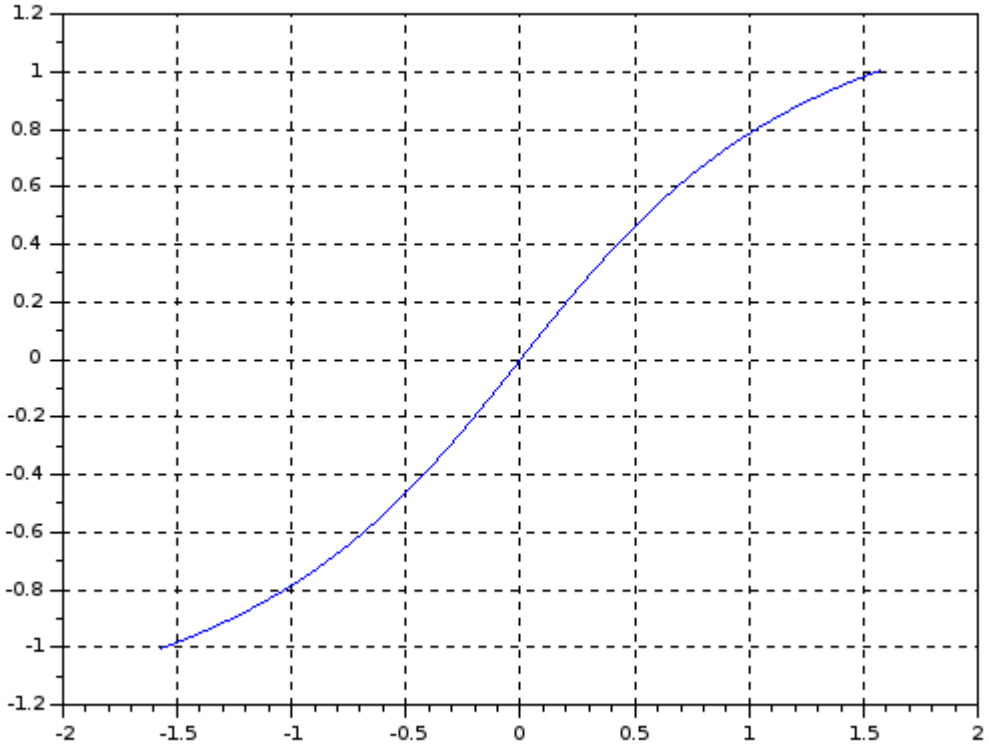
```
//dibujar un triangulo
P1=[5,3]
P2=[0,1]
P3=[6,2]
x=[P1(1),P2(1),P3(1),P1(1)]
y=[P1(2),P2(2),P3(2),P1(2)]

f3=scf(3)
plot(x,y,"k--")
```



```
//dibujar una función

//// x initialisation
x=[0:0.1:2*%pi]';
////simple plot
f5=scf(5)
plot(x,sin(x))

//dibujar cuadricula
xgrid()

//dibujar titulos
xtitle( 'My surface', 'X axis', 'Y axis')

/
```



```
/dibujar función en intervalo A y B
function y=MyTan(x)
    y=atan(x)
endfunction


a=-%pi/2; b=%pi/2
Npasos=100
paso=(b-a)/Npasos
x=zeros(Npasos+1)
y=zeros(Npasos+1)
for i=1:Npasos+1
    x(i)=a+(i-1)*paso
    y(i)=MyTan(x(i))
end


f4=scf(4)
plot(x,y)
xgrid()
```



Subplot

subplot(m,n,p) breaks the graphics window into an m-by-n matrix of sub-windows and selects the p-th sub-window for drawing the current plot. The number of a sub-window into the matrices is counted row by row ie the sub-window corresponding to element (i,j) of the matrix has number (i-1)*n + j.

Ejemplo

```
//subplots
t=-%pi:0.3:%pi;
subplot(211)
plot2d(t,sin(t))
subplot(212)
plot2d(t,cos(t))
```