M'hamed Bougara University - Boumerdès
Faculty of Sciences
Computer Science Department

Course: Programming Tools for Mathematics Matlab
L1

Dr RAHMOUNE Nabila

2023/2024

Chapter 3
Programming with Matlab

Outline

- Graphics
- Control structures
- Repetitive instructions
- M-File
- Functions
- Applications

Graphics

```
>> x = linspace(-1,1,10);
>> y = sin(x);
>> plot(x,y);     % plots y vs. x.
>> plot(x,y,'k-');   % plots a black line of y vs. x.
>>  hold on;     % put several plots in the same figure window.
>> figure;       % open new figure window.
```

Graphics

plot3(x,y,z) % plot 2D function.

mesh(x_ax,y_ax,z_mat) – surface plot.

contour(z_mat) – contour plot of z.

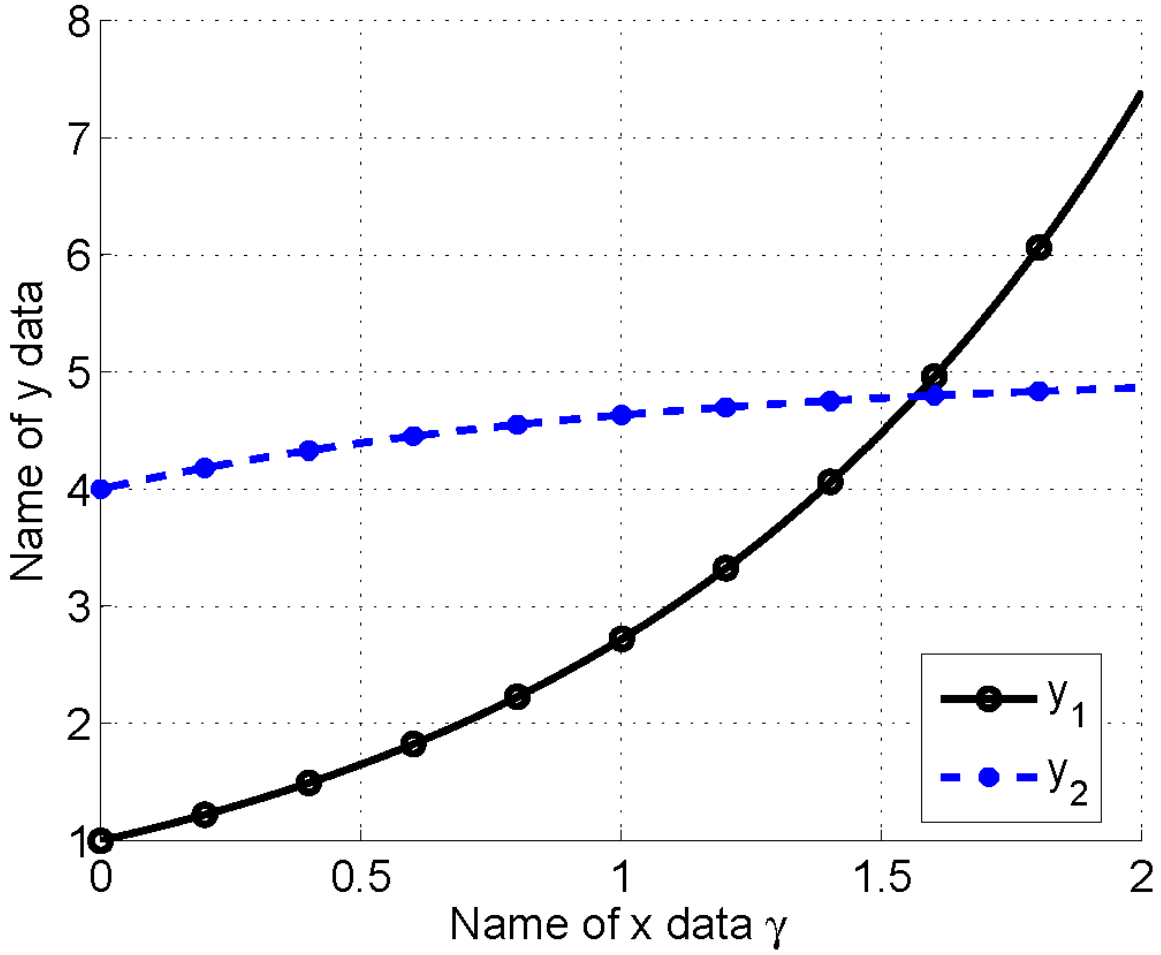
axis([xmin xmax ymin ymax]) – change axes

title(‘My title’); - add title to figure;

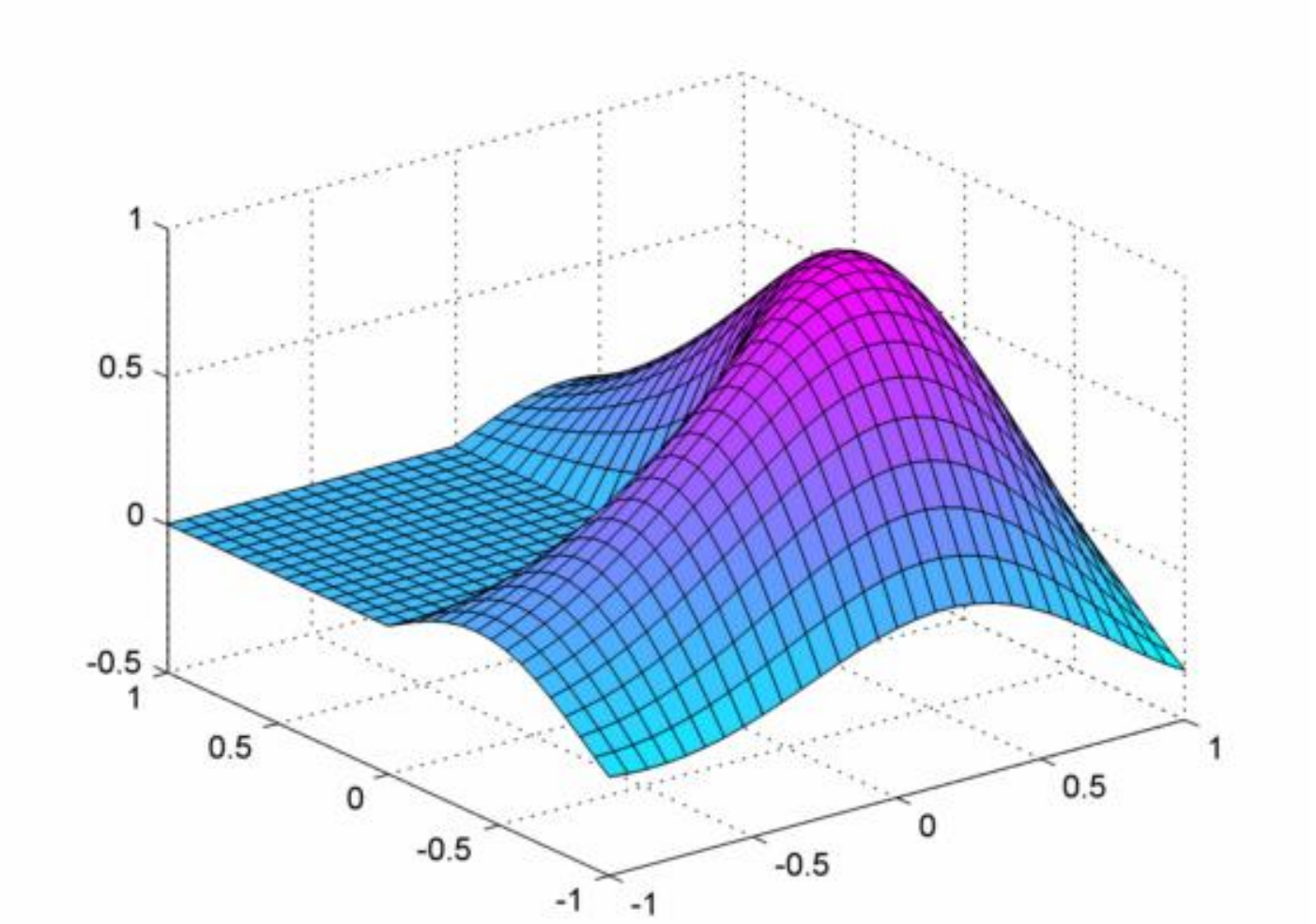
xlabel, ylabel – label axes.

legend – add key to figure.

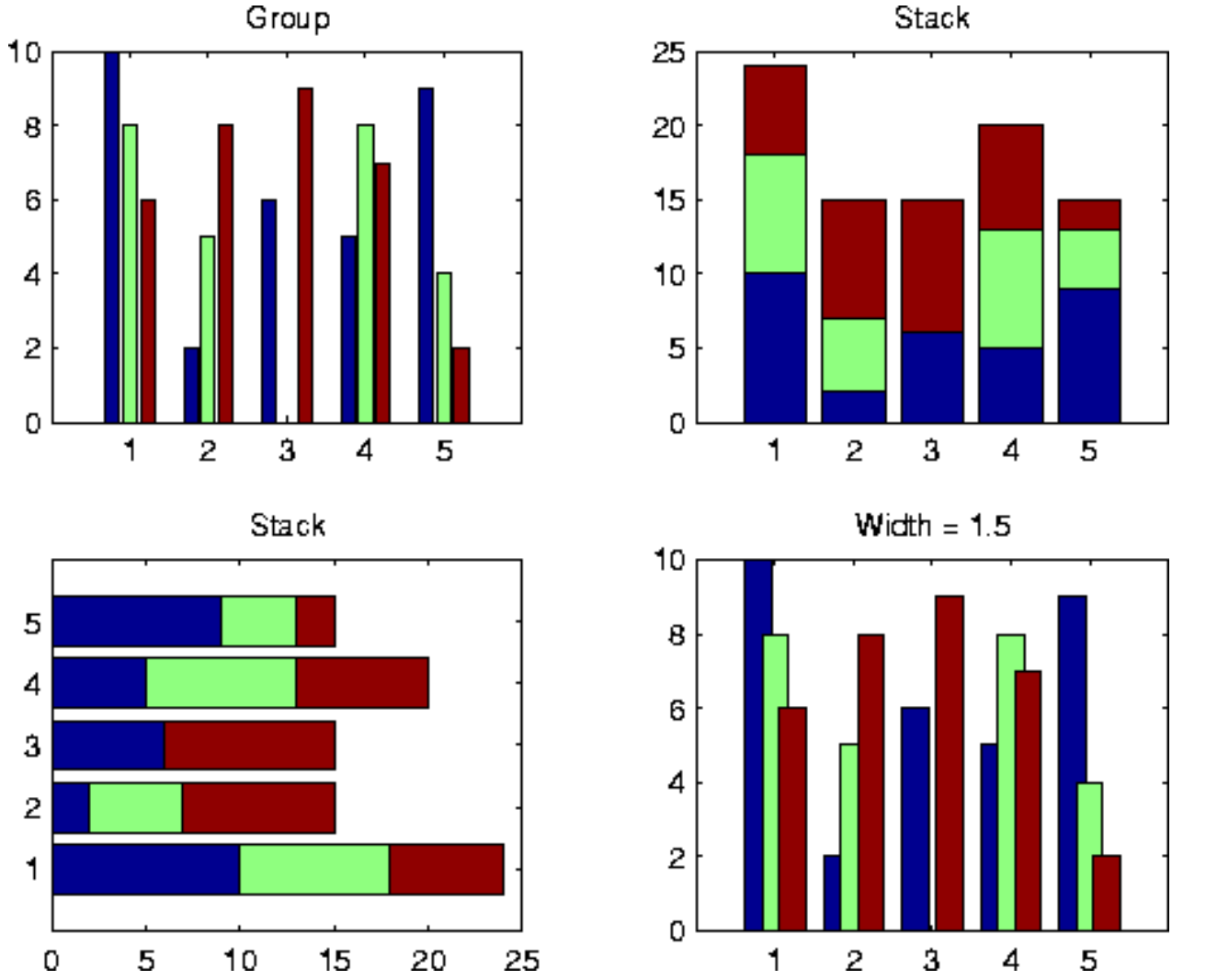
Examples of Matlab Plots



Examples of Matlab Plots



Examples of Matlab Plots



Plot the function sin(x) between 0≤x≤4π

Create an x-array of 100 samples between 0 and 4π.

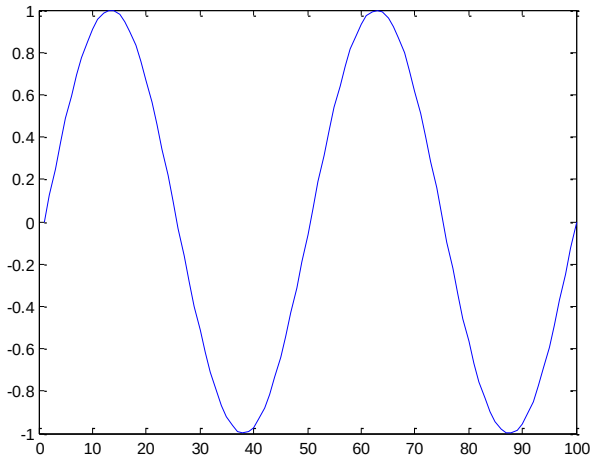
Calculate sin(.) of the x-array

```
>>x=linspace(0,4*pi,100);
```

Plot the y-array

```
>>y=sin(x);
```

```
>>plot(y)
```



Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

Create an x-array of 100 samples between 0 and 4π.

```
>>x=linspace(0,4*pi,100);
```

Calculate sin(.) of the x-array

```
>>y=sin(x);
```

Calculate $e^{-x/3}$ of the x-array

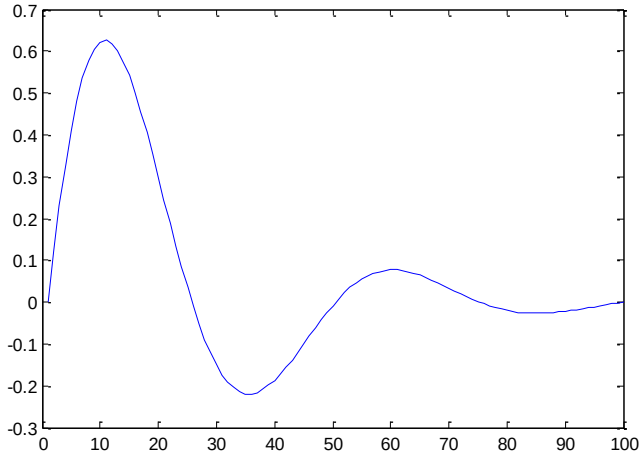
```
>>y1=exp(-x/3);
```

Multiply the arrays y and y1 correctly

```
>>y2=y.*y1;
```

Plot the y-array

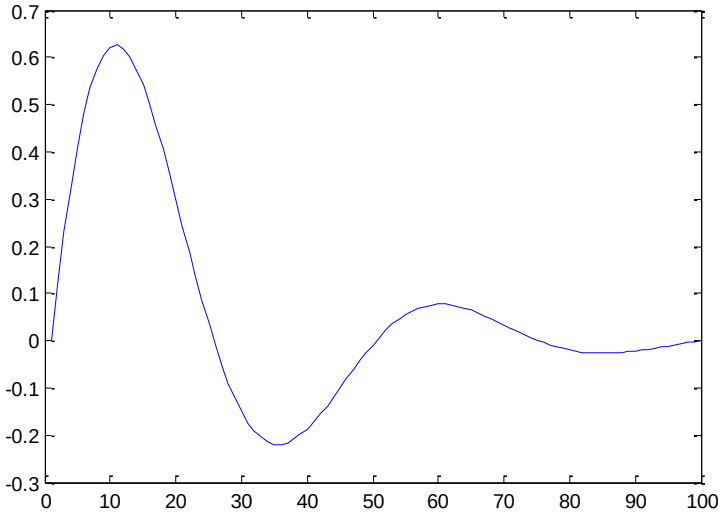
```
>>plot(y2)
```



Display Facilities

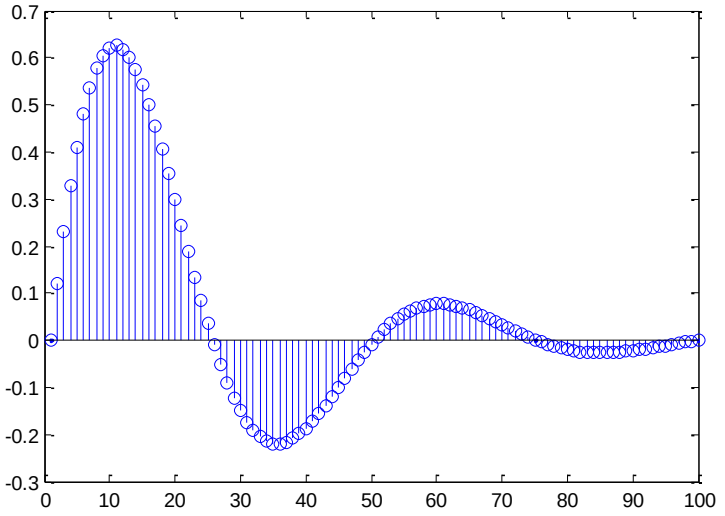
plot(.)

Example:
>>x=linspace(0,4*pi,100);
>>y=sin(x);
>>plot(y)
>>plot(x,y)



stem(.)

Example:
>>stem(y)
>>stem(x,y)



Display Facilities

title(.)

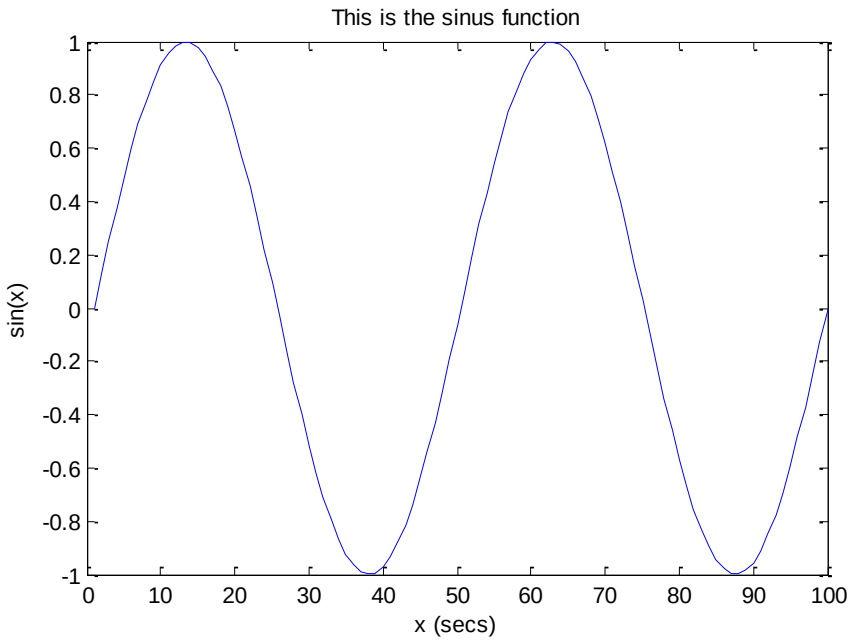
```
>>title('This is the sinus function')
```

xlabel(.)

```
>>xlabel('x (secs)')
```

ylabel(.)

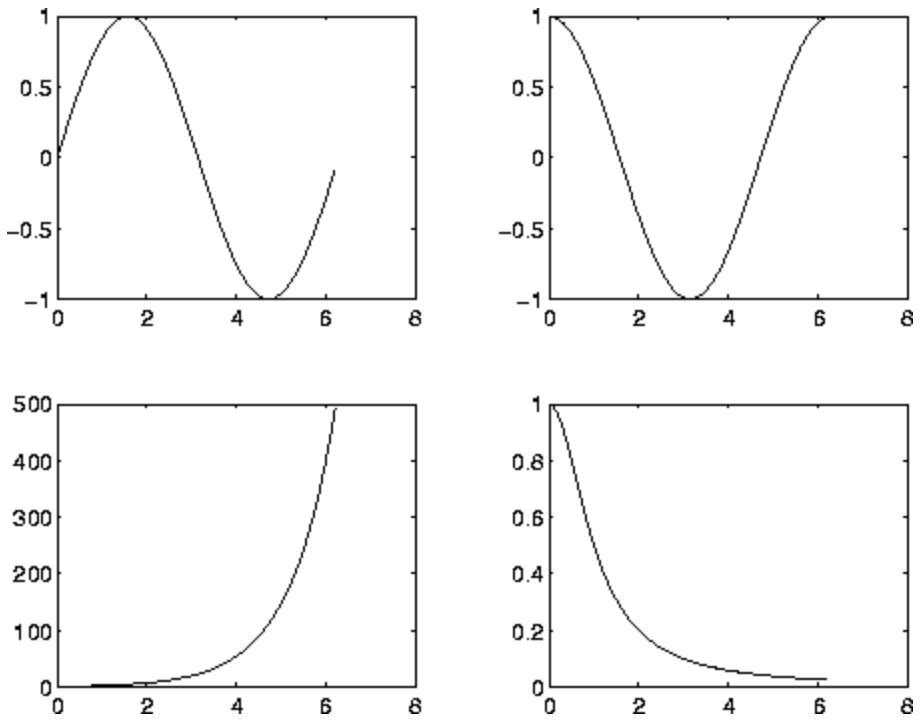
```
>>ylabel('sin(x)')
```



Putting several graphs in one window

The subplot command creates several plots in a single window. Here is an example:

```
>> t = (0:.1:2*pi)';
>> subplot(2,2,1)
>> plot(t,sin(t))
>> subplot(2,2,2)
>> plot(t,cos(t))
>> subplot(2,2,3)
>> plot(t,exp(t))
>> subplot(2,2,4)
>> plot(t,1./(1+t.^2))
```



Les fonctions hist et pie

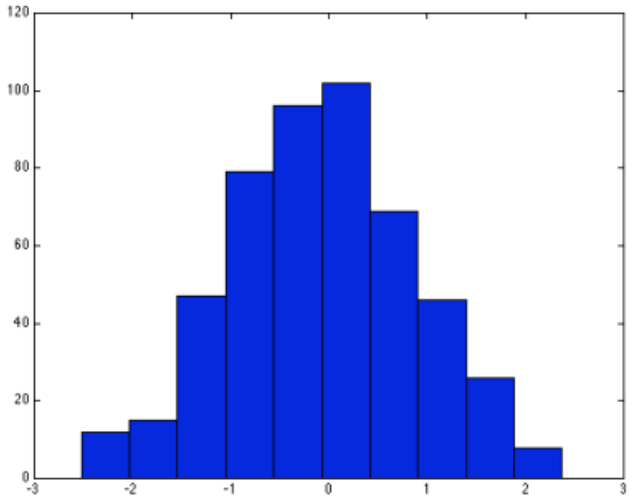
hist(y,n) : Draw a histogram that distributes the values of y into n classes.

Exemple

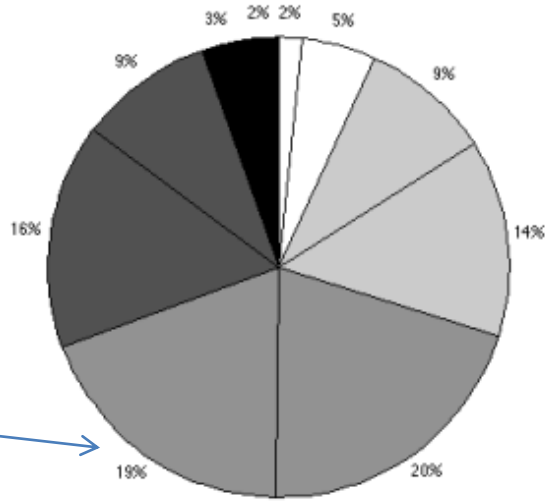
Let's create a vector of random numbers :

>> a = randn(1,500) ;

>> hist(a)



>> pie(N)

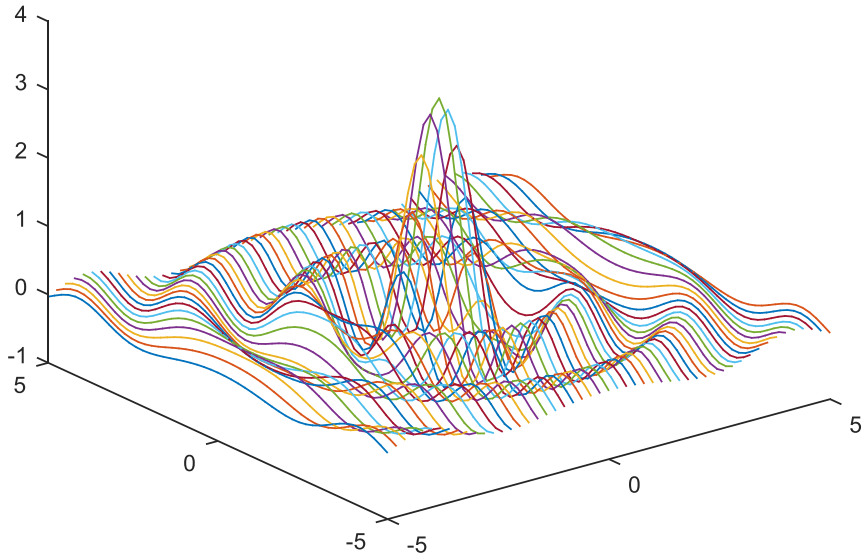


Space curves, function plot3

The plotting of lines in three dimensions is primarily done using the function plot3().

Example1

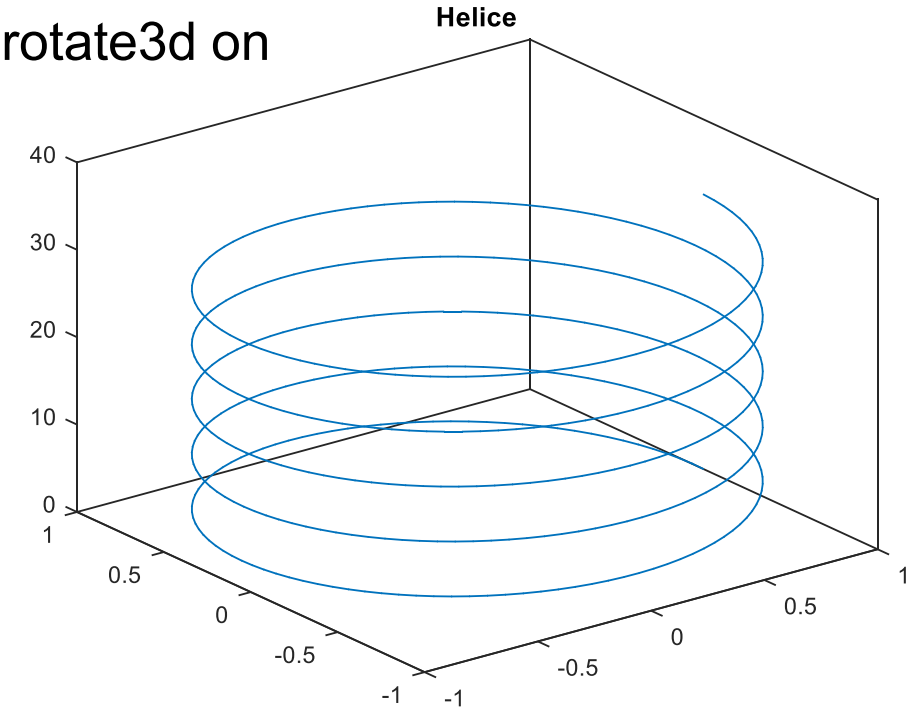
```
>> [x,y] = meshgrid(-5:0.2:5);
>> r = sqrt(x.^2 + y.^2) + eps;
z = sin(pi*r) ./ r;
close all;
plot3(x, y, z);
```



Space curves, function plot3

Example2

```
>> t = linspace(0,10*pi,500) ;
>> x = cos(t) ; y = sin(t) ;
>> plot3(x,y,t) ; title('Helice') ; box on ; rotate3d on
```

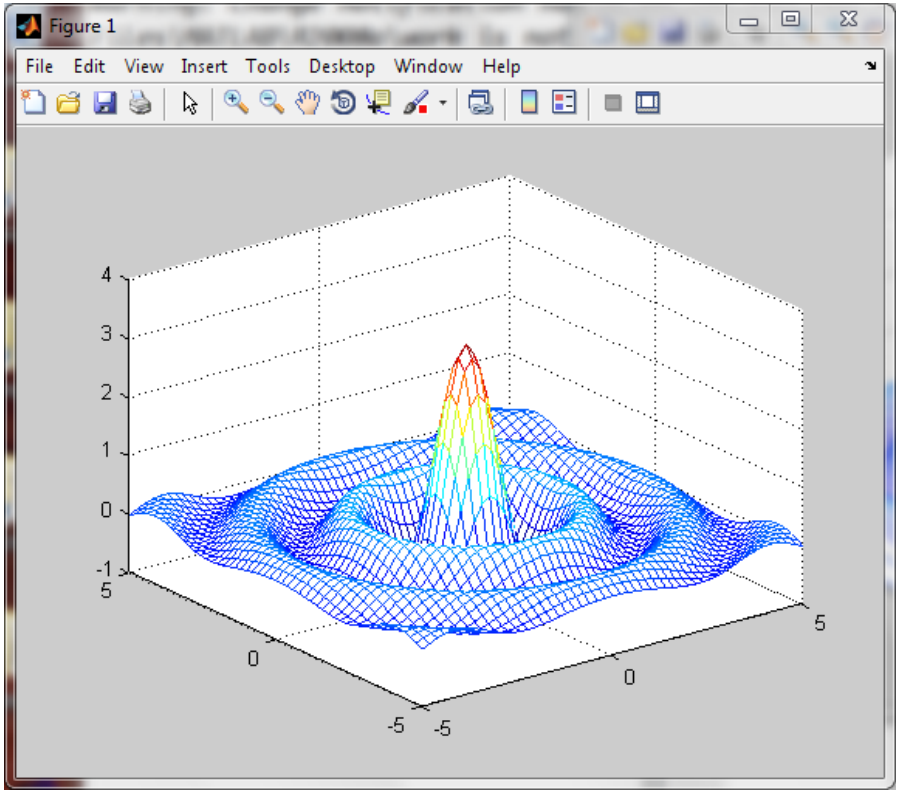


Space curves, function mesh

The function **mesh()** plots wireframe surfaces.

Example

```
>> [x,y] = meshgrid(-5:0.2:5);
>> r = sqrt(x.^2 + y.^2) + eps;
z = sin(pi*r) ./ r;
close all;
mesh(x, y, z);
```

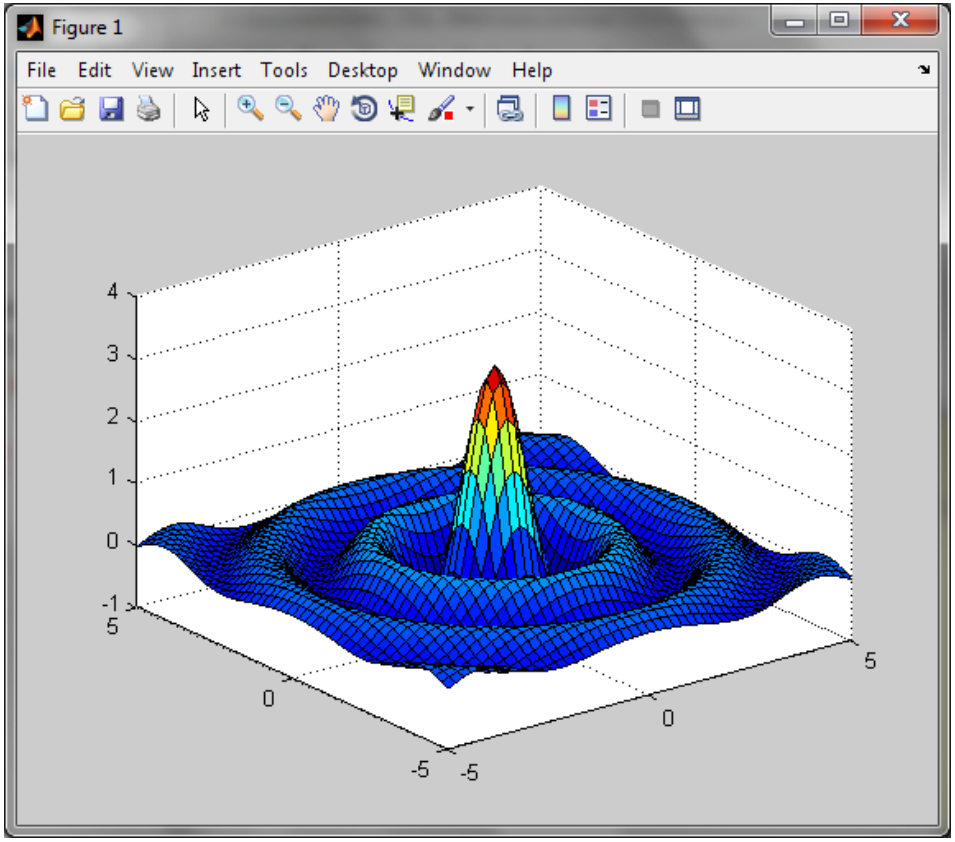


Space curves, function surf

The function **surf()** plots surfaces with more capabilities than the **mesh()** function.

Example

```
>> [x,y] = meshgrid(-5:0.2:5);
>> r = sqrt(x.^2 + y.^2) + eps;
z = sin(pi*r) ./ r;
close all;
surf(x, y, z);
```



Control Structures

If Statement Syntax

```
if (Condition_1)
                Matlab Commands
elseif (Condition_2)
                Matlab Commands
elseif (Condition_3)
                Matlab Commands
else
                Matlab Commands
end
```

Some Dummy Examples

```
if ((a>3) & (b==5))
    Some Matlab Commands;
end
```

```
if (a<3)
    Some Matlab Commands;
elseif (b~=5)
    Some Matlab Commands;
end
```

```
if (a<3)
    Some Matlab Commands;
else
    Some Matlab Commands;
end
```

Control Structures

For loop syntax

```
for i=Index_Array
        Matlab Commands
end
```

Some Dummy Examples

```
for i=1:100
    Some Matlab Commands;
end
```

```
for j=1:3:200
    Some Matlab Commands;
end
```

```
for m=13:-0.2:-21
    Some Matlab Commands;
end
```

```
for k=[0.1 0.3 -13 12 7 -9.3]
    Some Matlab Commands;
end
```

Control Structures

While Loop Syntax

```
while (condition)
        Matlab Commands
end
```

Example

```
while  ((a>3) & (b==5))
    Some Matlab Commands;
end
```

Construction switch...case

Syntaxe

```
switch (sélecteur)
case valeur 1, . . . / script 1 /
case Valeur 2, . . . / script 2 /
. . .
otherwise / script
end
```

Example

```
>> n =17;
>> switch rem(n,3)      % reste de la division de n par 3
       case 0, disp('Multiple de 3')
       case 1, disp('1 modulo 3')
       case 2, disp('2 modulo 3')
       otherwise, disp('Nombre négatif')
   end
```

Loop For nested

Example 2

```
>> for (l = 1 : 3)
        for ( k = 1 : 3)
                A(l,k) = l^2 + k^2 ;
        end
    end
```

```
>> disp(A)
02      05      10
05      08      13
10      13      18
```

Using break and continue

It is possible to exit directly from a for or while loop using the break command:

Example 3 (break)

```
>> epsi = 1;
>> for (n = 1 : 100)
      epsi = epsi / 2;
          If ((epsi + 1) <= 1)
              epsi = epsi*2
              break
          end
    .
```

The test (epsi + 1) <= 1 causes the for loop to exit at the 52nd iteration

```
epsi = 2.2204e-16
>> n
n = 52
```

Using break and continue

Example 4 (break)

In the following example, table A is that of example 2.

```
>> for (l = 1 : 3)
        for (k = 1 : 3)
            if (A(l,k) == 10)
                [l,k]
                break
            end
        end
    end
```

```
ans =                ans=
```

The double loop did not stop after the test A(l,k) == 10 was validated when l=1 and k=3. In fact, break causes the exit of the closest loop, here the internal for loop

La double boucle se s'est pas arrêté après que le test A(l,k) == 10 ait été validé lorsque l=1 et k=3. En effet break provoque la sortie de la boucle la plus proche, ici la boucle for interne.

Using break and continue

A corrected version of the previous test could be the following with two breaks to be able to exit the two for loops:

Example 5 :

```
>> sortie = 0;
>> for (l=1:3)
        if (sortie)
           break
        end
         for (k = 1:3)
              if (A(l,k) == 10)
                    [l,k]
                    sortie = 1 ;  break
              end
            end
   end
ans =     1     3
```

The continue command interrupts the execution of the body of the loop and causes a transition to the next iteration

La commande continue interromps l'exécution du corps de la boucle et provoque le passage à l'itération suivante

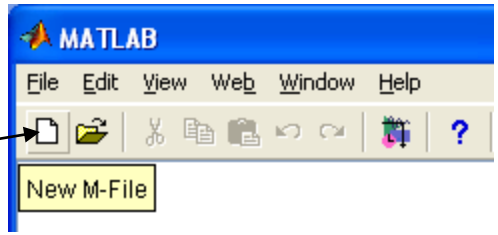
Example 6 :

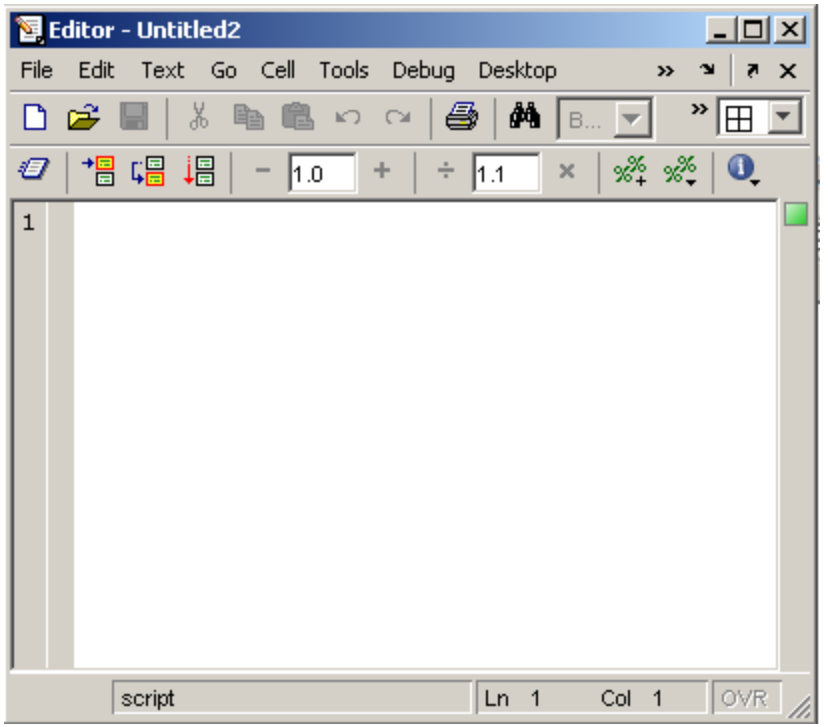
```
>>s=0 ;
>>for i=1:3
       if i==2
         continue
        else
          s=s+i;
        end
end

>> s=
     4
```

Use of M-File

Click to create
a new M-File

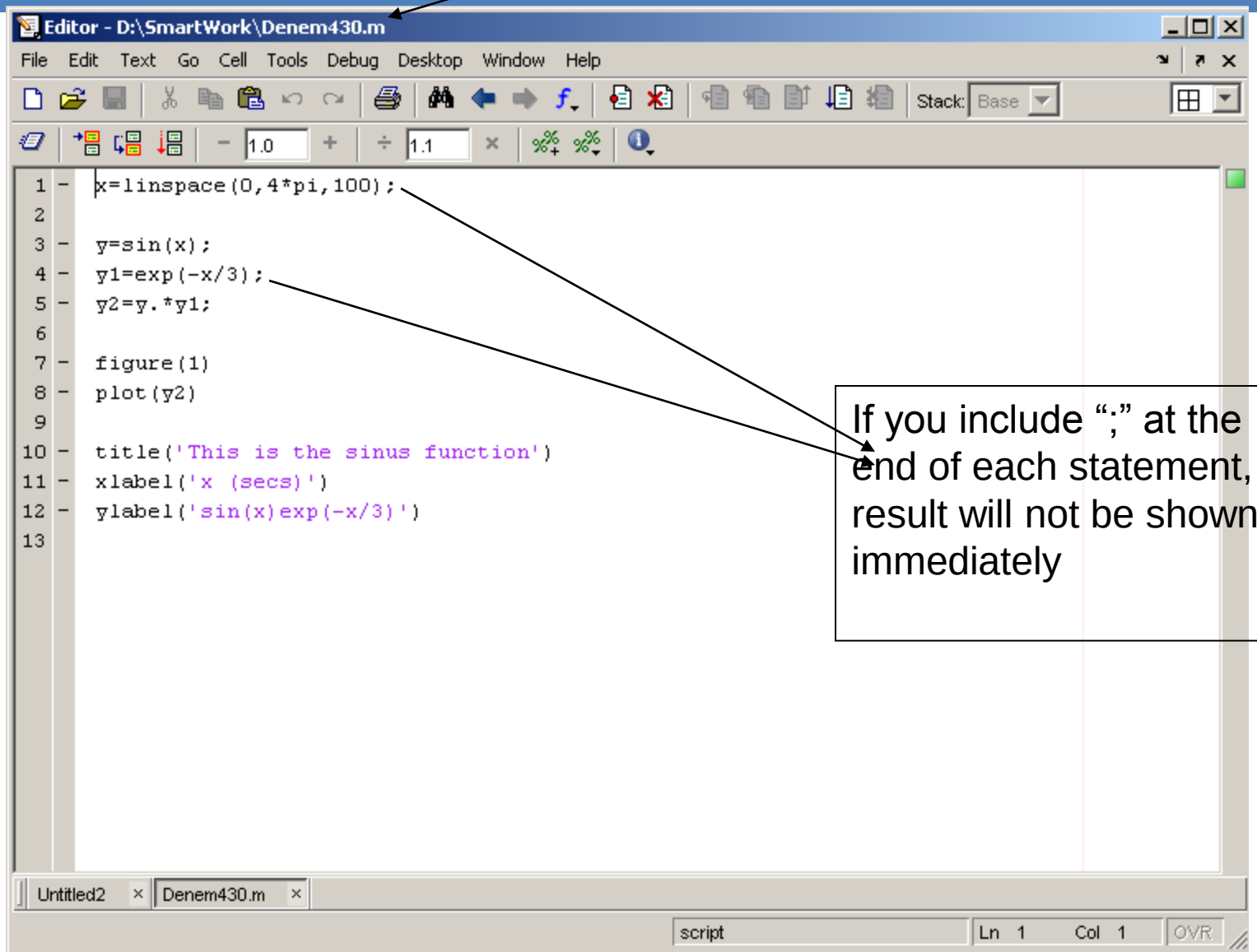




- Extension ".m"
- A text file containing script or function or program to run

Use of M-File

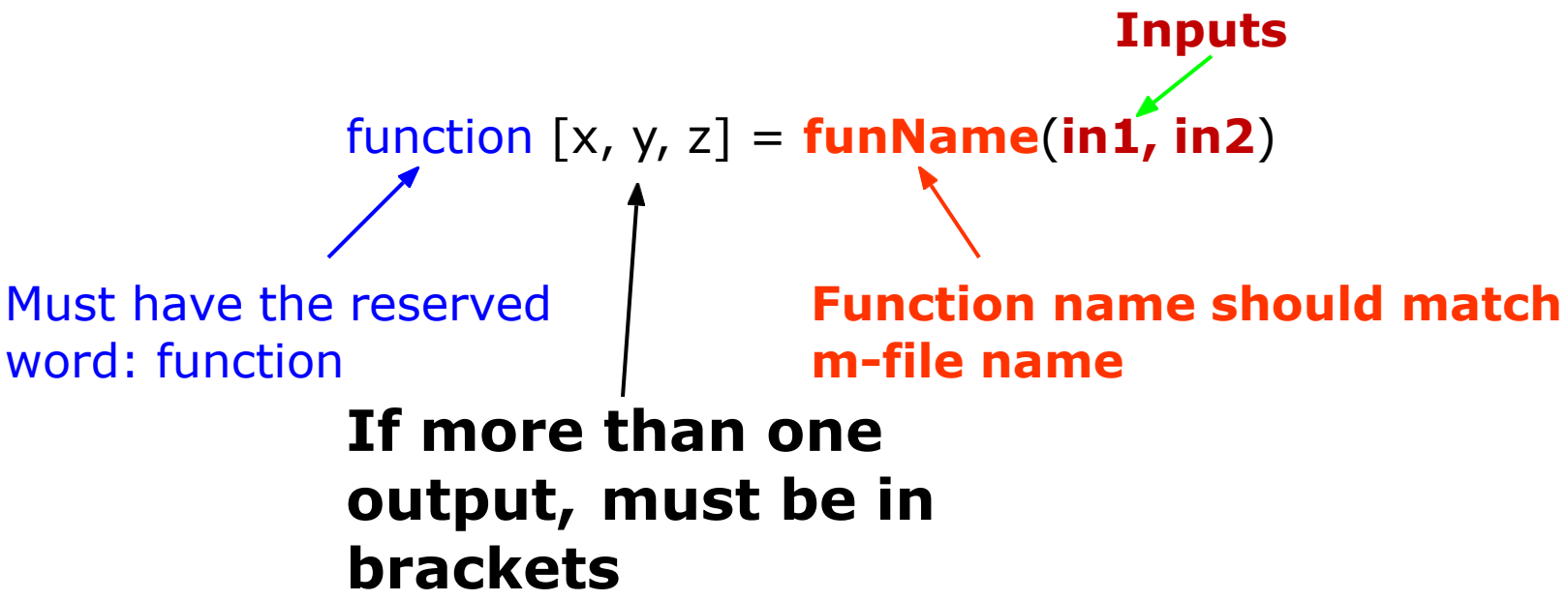
Save file as *Denem430*.m



If you include ";" at the end of each statement, result will not be shown immediately

Functions

- Some comments about the function declaration



- **No need for return:** MATLAB 'returns' the variables whose names match those in the function declaration
- **Variable scope:** Any variable created within the function but not returned disappears after the function stops running (They're called "local variables")

Writing User Defined Functions

Functions are **m-files** which can be executed by specifying some inputs and supply some desired outputs.
The code telling the Matlab that an m-file is actually a function is

```
function out1=functionname(in1)
function out1=functionname(in1,in2,in3)
function [out1,out2]=functionname(in1,in2)
```

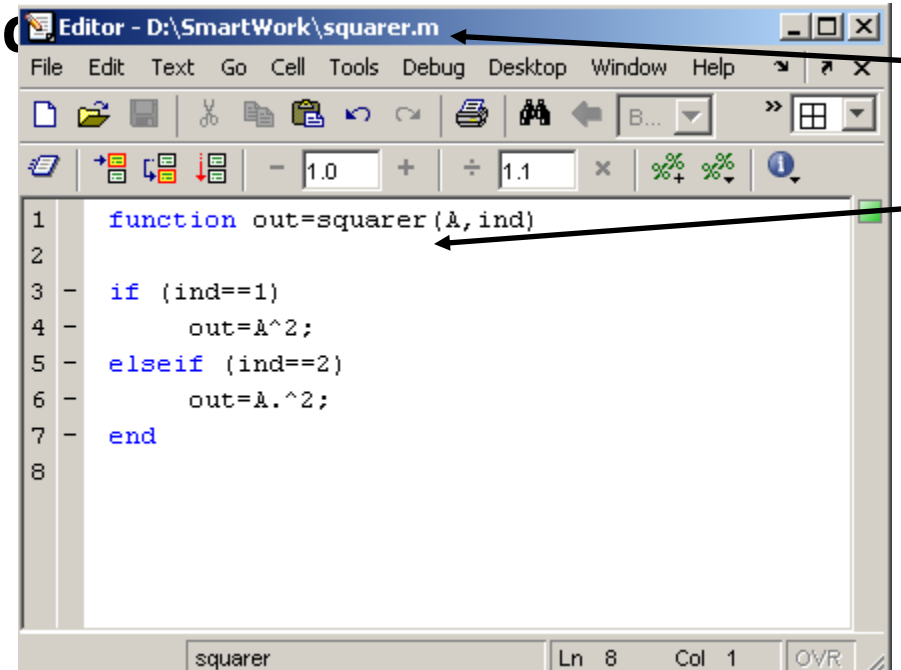
You should write this command at the beginning of the m-file and you should save the m-file with a file name same as the function name

Writing User Defined Functions

Examples

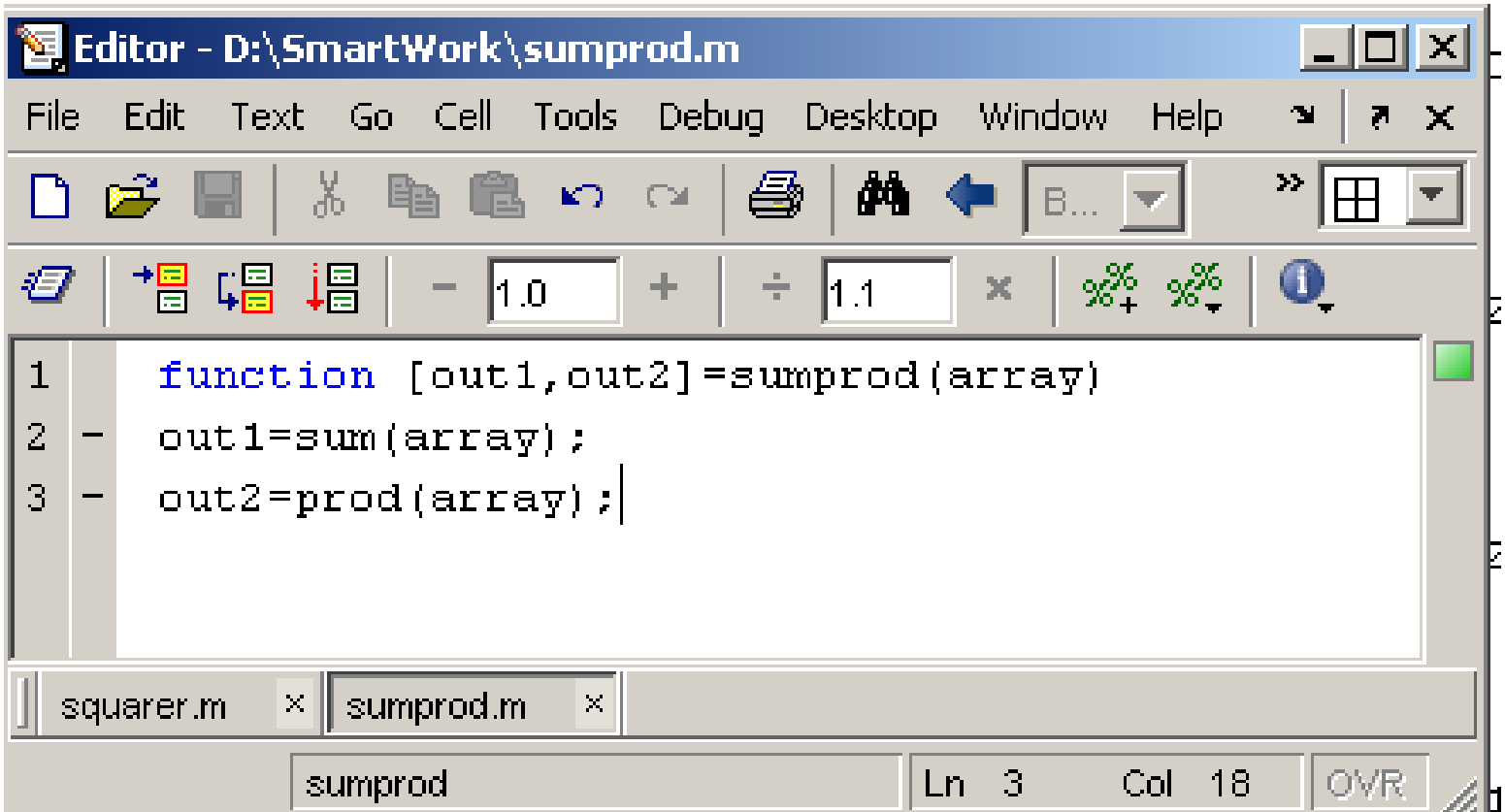
Write a function : out=squarer (A, ind)

Which takes the square of the input matrix if the input indicator is equal to 1 and takes the element by element square of the input matrix if the input indicator is equal to



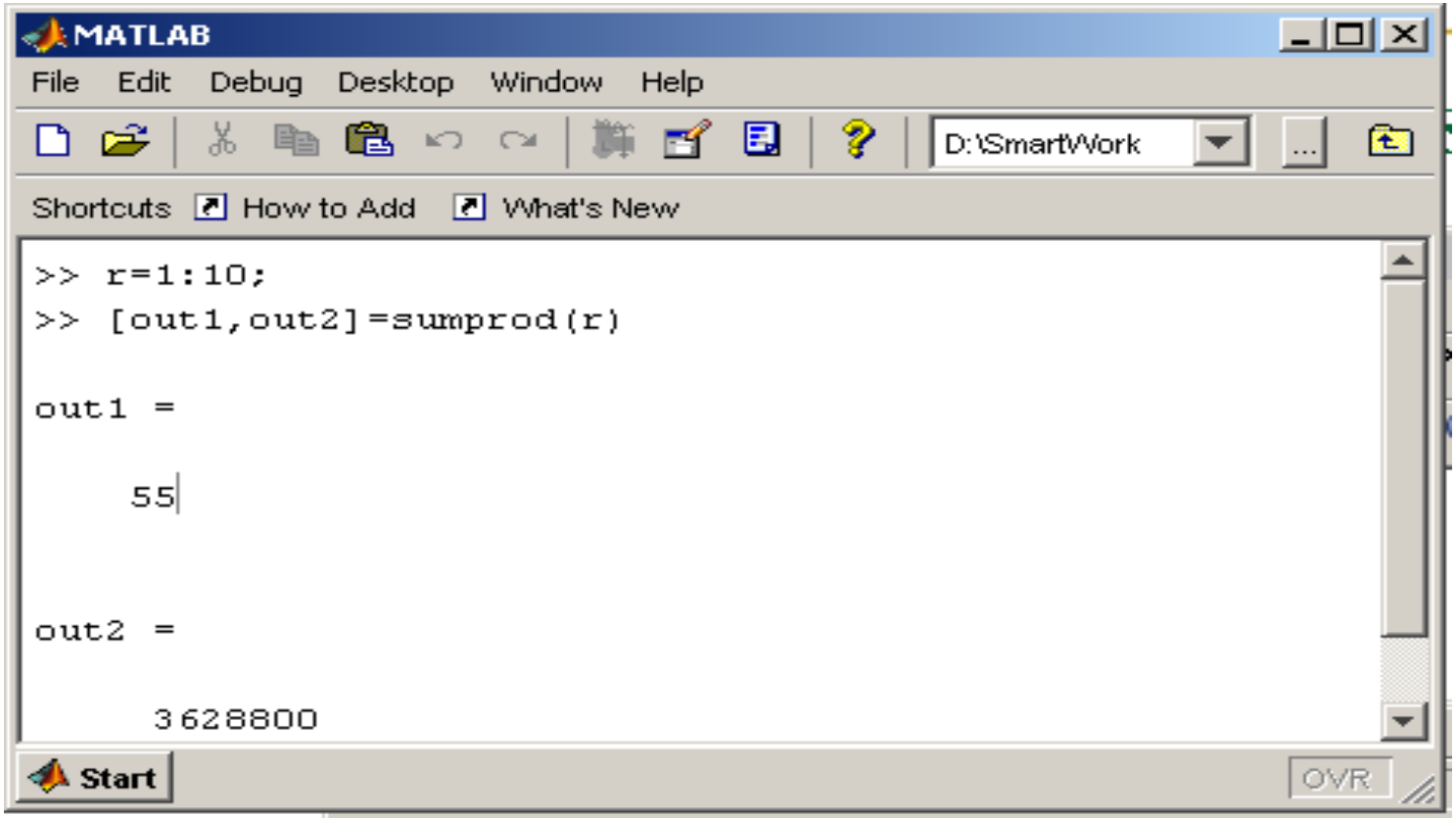
Writing User Defined Functions

Another function which takes an input array and returns the sum and product of its elements as outputs



Writing User Defined Functions

Function sumprod can be called from command windows



Writing User Defined Functions

```
%% square.m ---- Calculates the square of a number.


function y = square(x)
% calculate the square of the given number 'x'
%  Arguments:
% x (input)   value to be squared
% y (output)  the result of the square
  y = x*x;
  end
% end of square function
```

Notes

- “%” is the neglect sign for Matlab (equaivalent of “//” in C).
- Anything after it on the same line is neglected by Matlab compiler.
- Sometimes slowing down the execution is done deliberately for observation purposes.

You can use the command “pause” for this purpose

```
pause          % wait until any key
pause(3)       % wait 3 seconds
```

m-Functions (many functions in the same script)

% MYSTAT(X) : moyenne and variance of a elements's list are vector

```
function [m, v] = myStat(x)          % main function
   [k,l] = size(x) ;
   if ( (k~=1) & (l~=1) )
      error('l"argument must be  vector')
   end
  m = moyenne(x);
  v = variance(x);
 return
```

Example 2 : moyenne and variance (file : myStat.m)

```
function a = moyenne(u)
% Calcul de la moyenne
     a = sum(u)/length(u);
```

•The set of 3 functions is saved in one and same file m-file with main function name myStat.m.

Example : average and variance (file : myStat.m)

```
function [m, v] = myStat(x)
% MYSTAT(X) : average and variance of elements'liste array

[k,l] = size(x) ;
   if ( (k~=1) && (l~=1) )
      error('The argument must be a list or a vector. ')
   end
  m = moyenne(x);      %  function called
  v = variance(x);
 return

 function a = average(u)
 % Calcul of the average
      a = sum(u)/length(u);

 function b = variance(u)
  % Calcul of the  variance
   c = sum(u)/length(u);
   u2 = (u - c).^2;
   b = sum(u2)/length(u);
```

- **Set of three functions are saved in one file m-file with the main function's name myStat.m**

Application

Matlab programm that allows to calculate n! pour n = 1 à 100 with 2 methods and compare their execution times.

Method 1

```
tic
fact = zeros(1,100);
 n=100;
fact(1)=1;

for k=2:n
   fact(k)=fact(k-1)*k;
end
toc
```

Method2

```
tic
 n=100;
 fact=cumprod(1:n); toc
```

Prpgram of method 2 is faster then program of the first methos.

To see it, and compare the speed of the program, we can use **"tic" (to be placed at the beginning of the program) and "toc" (to be placed at the end of the program)**, which allow us to measure the elapsed CPU time.