UNIVERSITÀ DEGLI STUDI
DI MILANO

Informazione Multimediale

Raffaella Lanzarotti

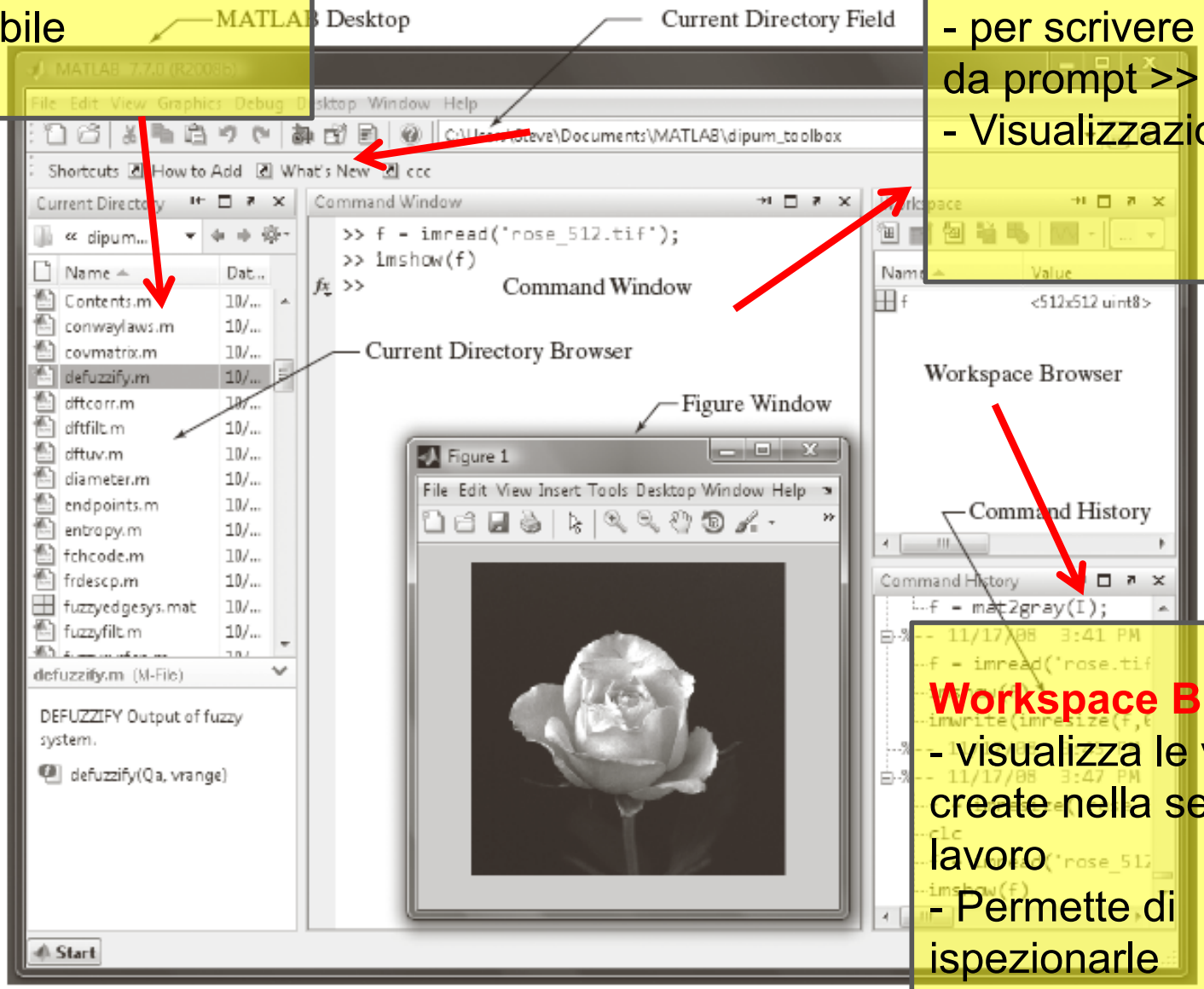
Matlab

- **MATLAB**: MA-trix LAB-oratory
- nato come programma per la *gestione di matrici*
- strumento di *analisi dati e di visualizzazione* largamente usato per attività di ricerca e simulazione.
- Vantaggi: facilità di utilizzo e intuitività delle funzioni.
- Svantaggi: utilizza pesantemente la memoria del computer, e i tempi di elaborazione sono molto elevati.

Desktop

Current directory:
-visualizza la directory corrente
- modificabile

Command Window:
- per scrivere comandi da prompt >>
- Visualizzazione output



Workspace Browser:
- visualizza le variabili create nella sessione di lavoro
- Permette di ispezionarle

FIGURE 1.1 The MATLAB Desktop with its typical components.

Prompt: >>

- Permette:
  - dichiarazioni di variabili
  - espressioni
  - chiamate a funzioni (di Matlab o meno)

- permette di richiamare gli ultimi comandi inseriti usando le frecce in alto e in basso.

Comandi di recupero informazioni

- On-line help
  - help *argomento*: fornisce l'help sull'argomento indicato
  - lookfor *argomento*: visualizza tutte le f. che hanno nella prima riga di commento (H1 line) la parola nell'argomento
  - helpwin: help interattivo completo

- Workspace information
  - who, whos: elenca le variabili correntemente in uso
  - clear: rimuove tutte le variabili in uso
  - clear x,y,z
  - clear all

- Directory information
  - pwd: mostra la directory di lavoro corrente
  - cd: cambia la directory di lavoro corrente
  - dir: mostra il contenuto della directory di lavoro corrente
  - ls: stesso comportamento di dir

Help

- *help nome_funzione*
  - informazioni sulla sintassi della funzione richiesta
- *help nome_toolbox*
  - help di tutte le funzioni di un toolbox:
- Per sapere quali sono i *toolbox* digitare:
  - *help*
- *Qual è il toolbox di Image Processing?*
  «Images»
- lookfor *keyword* se non so il nome di una funzione, cerco le f. attinenti alla keyword

Comandi da prompt

- Es.1:

```
>> pi
ans =
   3.1416
>>
```

Matlab crea una variabile ***ans*** a cui assegna il valore richiesto

- Es.2:

```
>> help if
IF Conditionally execute statements.
    The general form of the IF statement is

       IF expression
         statements
       ELSEIF expression
         statements
       ELSE
         statements
       END
...
```

Esempi di comandi

Grandezze Scalari

```
>> 13*9

ans =

   117

>> sqrt(167)

ans =

   12.9228

>> sin(pi/6)

ans =

    0.5000

>> i

ans =

         0 + 1.0000i
```

La variabile **ans** viene utilizzata da MATLAB per la memorizzazione del risultato più recente non assegnato ad altre variabili

i e **j** se non assegnate rappresentano l'unità immaginaria

Esempi di assegnazione e operazioni tra scalari

```
>> x=5.5

x =

        5.5000

>> y=x^2

y =

       30.2500

>> z=y-x

z =

       24.7500
```

Matrici e vettori

- **Matrici e vettori**: valori tra parentesi quadre […],
- La **virgola** o lo spazio: separa gli elementi sulla riga
- Il **punto e virgola** o ENTER: nuova riga

- Es. di vettore riga: x = [1, 2, 3];
- Es. di vettore colonna: y = [1; 4; 7];

- Es. di matrice: A = [1 2 3; 4 5 6; 7 8 9];
- oppure: A = [1 2 3
  4 5 6
  7 8 9];

Esempi di assegnazione a vettori/matrici

```
>> a=[1 2 3 4 5 6]
a =
     1     2     3     4     5     6


>> b=[1 ; 2 ; 3 ; 4 ; 5 ; 6]
b =
     1
     2
     3
     4
     5
     6

>> A = [1 2 3; 4 5 6; 7 8 9]

A =
     1     2     3
     4     5     6
     7     8     9
>>
```

Operazioni fondamentali

```
>>B = [
1  2  3
4  5  6
7  8  9]

B =

     1     2     3
     4     5     6
     7     8     9
```

Si può utilizzare il tasto ENTER anche se l'espressione non è ancora completa, ma MATLAB non la esegue fino al suo completamento

```
>> isequal(A,B)

ans =

     1
```

Funzione Booleana: restituisce 1 se le variabili sono uguali, 0 in caso contrario

```
>> size(A)

ans =

     3     3
```

Restituisce le dimensioni della matrice, nell'ordine di righe e colonne

Operazioni fondamentali

```
>> A(2,3)

ans =

     6

>> A(:,2)

ans =

     2
     5
     8

>> A'

ans =

     1     4     7
     2     5     8
     3     6     9

>>
```

- Ordine degli indici di una matrice (img):

  (ind-righe , ind-colonne)

- Il carattere "**:**" viene utilizzato per indicare un intero intervallo (tutte le righe in questo caso)

- " ' " : Matrice Trasposta

Operazioni fondamentali

```
>> inv(A)
ans =
  1.0e+016 *
    0.3152   -0.6304
0.3152
   -0.6304    1.2609   -
0.6304
    0.3152   -0.6304
0.3152
```

- **Matrice Inversa**: può restituire errore nel caso di matrici singolari o quasi singolari

```
>> A*B
ans =
    30    36    42
    66    81    96
   102   126   150
```

- **Prodotto Righe per Colonne** tra due matrici (ci deve essere accordo di dimensioni)

```
>> A.*B
ans =

     1     4     9
    16    25    36
    49    64    81
```

- **Prodotto Puntuale**: si ottiene premettendo il carattere **.** all'operazione da realizzare. L'operazione viene effettuata elemento per elemento

Operazioni fondamentali

```
>> A^2

ans =

    30    36    42
    66    81    96
   102   126   150

>> A.^2

ans =

     1     4     9
    16    25    36
    49    64    81

>>
```

```
>> C = [1 2 3; 4 5 6]

C =

     1     2     3
     4     5     6

>> C^2
??? Error using ==> mpower
Matrix must be square.

>> C.^2

ans =

     1     4     9
    16    25    36

>>
```

- **Potenza su una matrice** solo su matrici quadrate

- Utilizzando l'**operatore puntuale** si ottiene l'operazione elemento per elemento

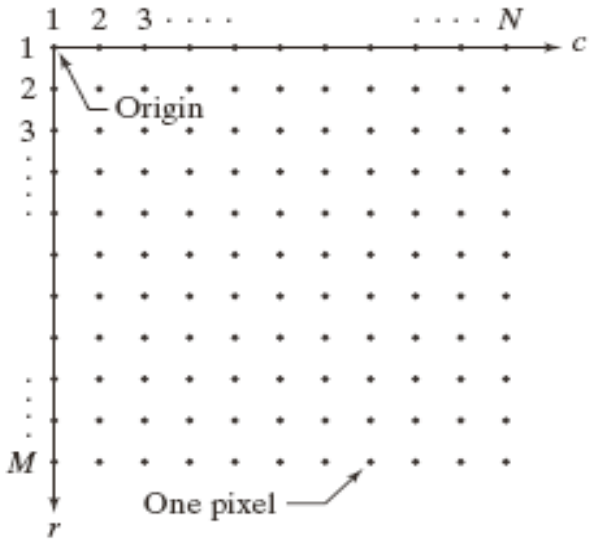
Vettori e matrici

In MATLAB è importante ricordare che

1. tutti i vettori sono indicizzati **partendo da 1**.

x(1)

2. Le matrici hanno indici (1:M, 1:N) dove
   - M = num. righe
   - N = num. colonne



Notazione a colonna

Il comando : può essere usato per generare vettori:

- senza specificare incremento

  es. `t=1:5` => `t=[1 2 3 4 5]`

- con incremento positivo specificato

  es. `t=0:0.2:1` => `t=[0 0.2 0.4 0.6 0.8 1]`

- con incremento negativo specificato

  es. `t=2:-0.2:1` => `t=[2 1.8 1.6 1.4 1.2 1]`

Matrici

- E’ possibile costruire matrici automaticamente:

```
>>  a=zeros(2)
a =
     0     0
     0     0
>>
```

- mentre:

```
>> a=zeros(2,3)
a =
     0     0     0
     0     0     0
>>
```

- Funzioni di Matlab (come zeros) accettano un numero variabile di elementi in input.
  - ones (genera matrici di 1)
  - rand (genera matrici di numeri casuali)
  - eye (genera le matrici identità)

Operazioni su matrici

operazioni algebriche sulle matrici (+ - * / ^):

Es:

```
>> a=ones(2,3);
>> b=ones(2,3);
>> a+b
ans =
     2     2     2
     2     2     2
>>
```

o anche:

```
>> a=2*eye(2)
a =
     2     0
     0     2
>>
```

Operazioni su matrici (cont.)

Altri opertori (.*, ./, .^) :
permettono di fare operazioni elemento per elemento senza ricorrere a cicli

Es:

```
>> x=[-1,2];
>> x^2
??? Error using ==> mpower
Matrix must be square.

>> x.^2
ans =
     1     4
```

o anche:

```
>> y = sin(x) .* cos(x);
```

Funzioni su matrici

- Funzioni operanti essenzialmente su vettori (riga o colonna) sono:
  - max, min,
  - sort,
  - sum, prod

- Funzioni operanti essenzialmente su matrici

inv	←	**inversa della matrice**
det	←	**determinante della matrice**
size	←	**dimensioni della matrice**
rank	←	**rango della matrice**

Funzioni su matrici

Per manipolare **l'orientamento** delle matrici si possono utilizzare le funzioni:

```
>> flipud(A)
ans =
     7     8     9
     4     5     6
     1     2     3

>> fliplr(A)
ans =
     3     2     1
     6     5     4
     9     8     7

>> rot90(A)
ans =
     3     6     9
     2     5     8
     1     4     7
```

Operatori relazionali

Sono utilizzabili in MATLAB i seguenti operatori relazionali

- == : uguale
- ~= : diverso da
- < : minore di
- <= : minore o uguale

Esempio:

```
>> x = 2;
>> x == 0

ans =

     0
```

Operatori relazionali

Possono essere applicati anche alle matrici:

```
>> a=[1 2; 0 -1];
>> a > 0

ans =

     1     1
     0     0
>> a >= 0

ans =

     1     1
     1     0

>>
```

Istruzione if-then-else

Sintassi:

```
if condizione
    operazioni
end
```

Utilizzo di **else e elseif** (opzionali)

```
if condizione1
    operazioni1
elseif condizione2
    operazioni2
else
    operazioni3
end
```

Condizioni logiche

`Condizione1,2` **devono essere condizioni che restituiscono come risultato VERO o FALSO. Gli operatori disponibili per tali confronti sono:**

`<, >`

`<=, >=`

`==` ← uguale

`~=` ← diverso

`&` ← and logico

`|` ← or logico

`~` ← not logico

Operazioni

`Operazioni1,2,3` **sono le operazioni da compiere se la condizione corrispondente risulta vera. Le varie istruzioni sono separate da virgole e l'ultima è seguita da un punto e virgola.**

```
if       n==10,
            a=b*c,
            d=e/f;
elseif   n~=20,
            a=e*f,
            d=a/b;
else
            disp ('Errore !!!');
end;
```

Ciclo For

Sintassi:

```
for k = 1:step:n
    operazioni
end
```

Il ciclo esegue le operazioni incrementando la variabile k da 1 a n con il passo indicato in step (default =1).

Es:

```
>> s=0;
>> for i=1:10
       s=s+i;
   end
>> s
s =
    55
>>
```

calcola la somma dei primi 10 numeri interi

Cicli annidati

Es

```
>> n=4;
>> for i=1:n
     for j=1:i
          a(i,j) = 1;
     end
   end
```

crea una matrice triangolare inferiore:

```
>> a
a =
1  0  0  0
1  1  0  0
1  1  1  0
1  1  1  1
```

Cicli annidati

Es.:

```
>> a = zeros(k,k)   % Preallocate matrix
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n -1);
    end
end

>> a

a =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
```

Cicli for

L'utilizzo dei cicli for, applicato su Matrici in MATLAB, è particolarmente pesante per l'elaborazione.

Esempio: utilizzo della funzione *tic toc*

```
>> tic; for i = 1:10^6, sin(i);end; toc;

Elapsed time is … seconds.

>> tic; i=1:10^6; sin(i); toc;

Elapsed time is … seconds.
```

➔ **Usarli il meno possibile, cercando altre soluzioni**

Cicli While

Il ciclo while ha la seguente struttura

```
while condizione
    operazioni
end
```

Esempio

```
>> i=1;
>> while i<5
        i=i+1;
   end
>> i
i =
     5
```

Text editor

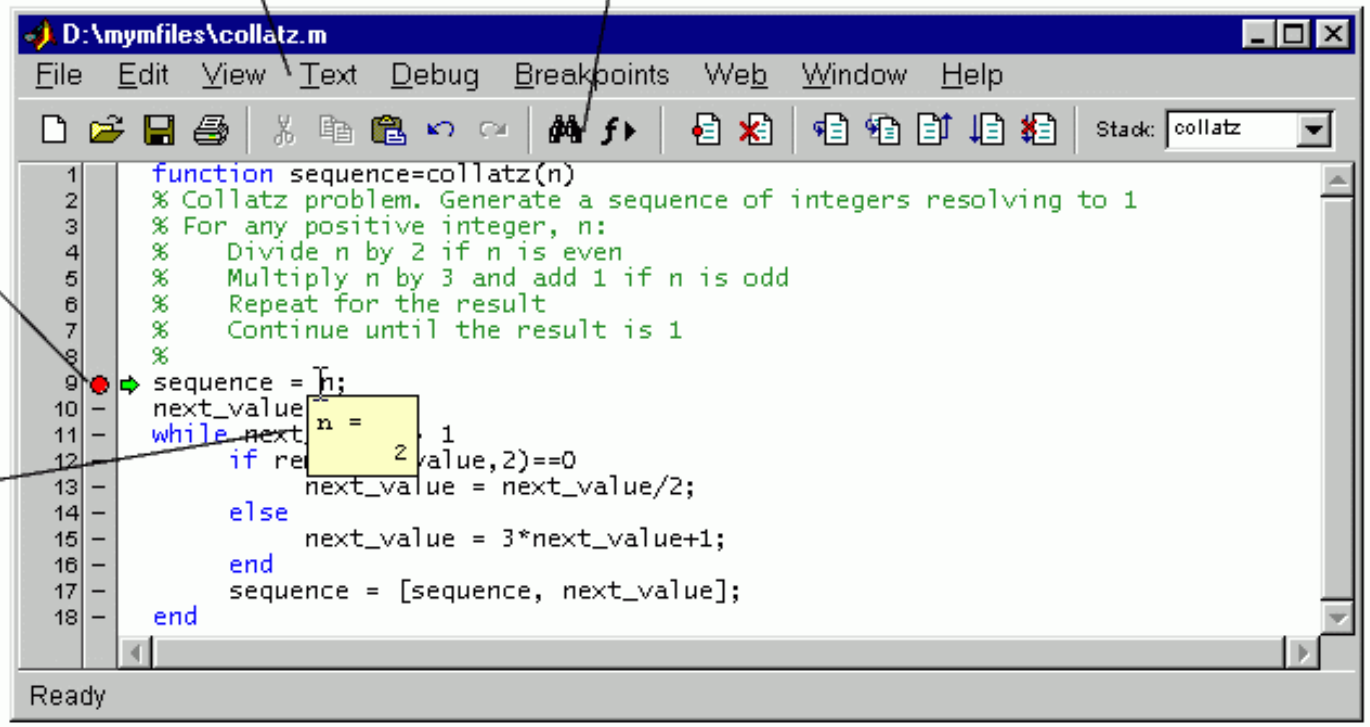
- Ambiente di editing degli M-files (script e funzioni)
- Debugger grafico di MATLAB

Commenta le linee selezionate e specifica lo stile di formattazione nel menu Text

Trova e sostituisci stringhe

Impostando Breackpoints si può mettere in pausa i programmi durante l'esecuzione e valutarne l'andamento

Tenendo il cursore su una variabile viene mostrato il suo valore attuale



File di MATLAB

- file di testo ASCII con estensione .m ;
- generati con un text-editor
- eseguiti digitando il nome sulla linea di comando (senza estensione!).
- Commenti: linea di commento col percento %

Script e funzioni

- **Scripts M-files**: sono files di comandi. Non hanno variabili in entrata e in uscita e operano sulle variabili del workspace

- **Funzioni M-files**: sono files di comandi, che hanno argomenti in entrata e in uscita. Le variabili interne a questi programmi non influenzano le variabili del workspace

NB: Non assegnare a nessuno script o funzione il nome di funzioni predefinite o il nome di una variabile usata!!!

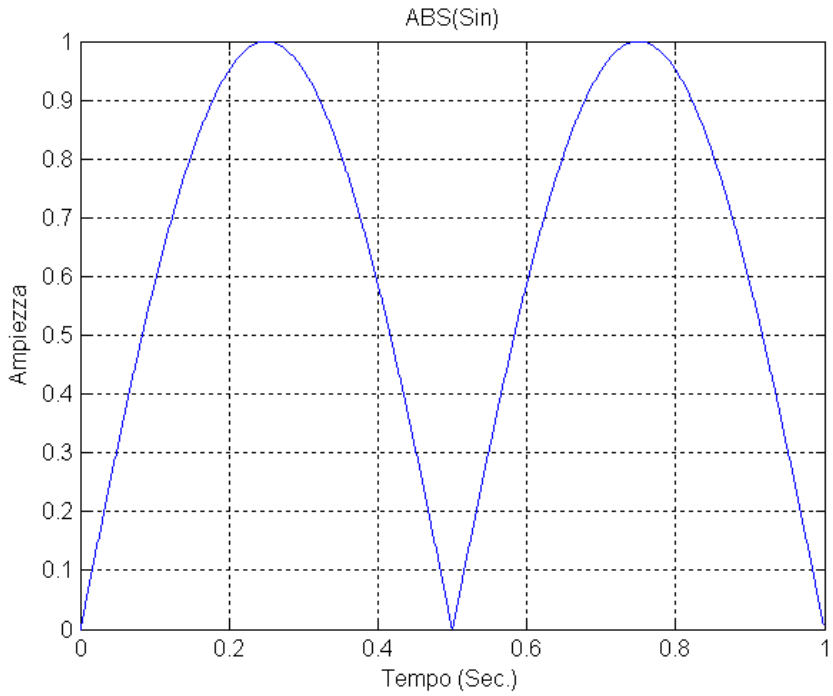
Esempio Script

```
% Script di esempio TSES_01_script.m
% Disegno del valore assoluto della funzione sin(2*pi*freq*t)
% su mille campioni equispaziati nell'intervallo t = [0,1).

Tmax = 1; % Durata
samples = 1000; % Numero campioni
delta = Tmax / samples; % DeltaT
freq = 1;
t = 0:delta:Tmax-delta;
s = abs(sin(2*pi*freq*t));

% equivale al ciclo:
% for i = 1:samples;
% t(i) = (i-1) * delta;
% s(i) = abs(sin(2*pi*freq*t(i)));
% end;

plot (t,s);
grid on;
title('ABS(Sin)');
xlabel('Tempo (Sec.)');
ylabel('Ampiezza');
```



Funzioni

Funzione: si crea come uno script – ma concettualmente molto diversa!!!

Il file contenente la funzione DEVE avere estensione .m e DEVE avere lo stesso nome della funzione

La funzione può ricevere e restituire dei risultati. Il passaggio dei parametri avviene per valore

IMPORTANTE: le variabili definite all'interno della funzione hanno visibilità locale e NON possono essere riferite dall'ambiente chiamante

Uso della dichiarazione global per l'utilizzo delle variabili nel Workspace su tutti i programmi

Es di funzione

```
function [xmin,xmax]=minmax(a,m,n)

%MINMAX(A,M,N) calcola l'elemento minimo, XMIN, e
%l'elemento massimo, XMAX della matrice a con m righe e
%n colonne

for i=1:m
  for j=1:n
    if a(i,j) > xmax
      xmax = a(i,j);
    end
    if a(i,j) < xmin
      xmin = a(i,j);
    end
  end
end
```

- E' corretta?
- Se no, dov'è l'errore?
- Come si può migliorarla passando solamente la matrice **a**?

Funzione corretta e ottimizzata

```
function [xmin,xmax]=minmax(a)

xmin= Inf;
xmax=-Inf; %inizializzazione!!!!

for i=1:size(a,1)
  for j=1:size(a,2)
    if a(i,j) > xmax
      xmax = a(i,j);
    end
    if a(i,j) < xmin
      xmin = a(i,j);
    end
  end
end
```

- Mancava l'inizializzazione
- Inoltre n e m si ricavano dalla matrice «a»
- Esistono f. MATLAB che calcolano direttamente il min e il max di una matrice?

Dichiarazione di una funzione

```
function [out1,out2,…]=nome_funzione(in1,in2,….)
```

- **Argomenti in output:** a sinistra dell' =, fra [ ]
- **Argomenti in input**: a destra dell' = , fra ( )
- Posso usare un numero di argomenti minore di quello indicato nella definizione della function, sia in entrata che in uscita (verranno assegnati i primi n).
- Es.: a=nome_funzione (b), assegna a "in1" il valore "b", e ad "a" il valore "out1"
- NB: controlla par in ingresso con nargin

Stringhe di testo, input, output

Il testo in MATLAB viene inserito sempre tra apici:

Es.: `string='Ciao';`

Per visualizzare stringhe o messaggi si adopera la funzione disp.

Es.: `disp('Premere un tasto');`

Funzione «error»

La funzione error mostra un messaggio di errore ed interrompe l'esecuzione di un file .m

Es.:
```
error('A deve essere simmetrica');
```

La funzione input mostra un messaggio e permette l'inserimento di dati.

Es.:
```
num_di_iter=input('Inserire il numero
                  di iterazioni: ');
```

Grafici

Passi:

1. Preparare un vettore di ascisse
2. Preparare un vettore di ordinate
3. Fare il grafico

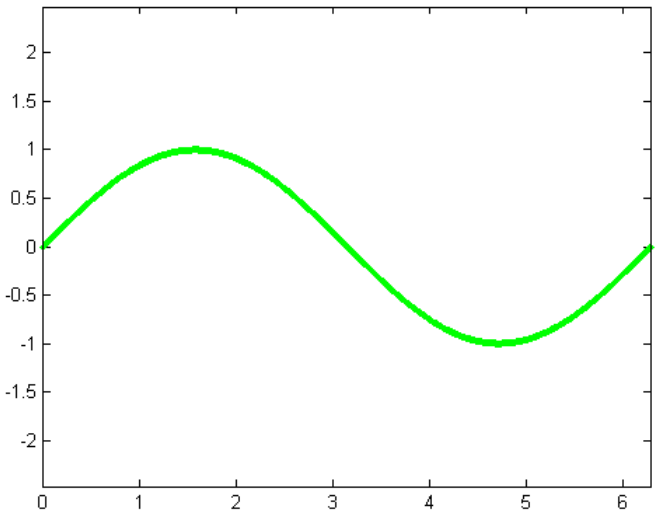
Esempio: grafico di cos(4x)*exp(x), su [0,2]

```
>> x=0:0.01:2;
>> f=cos(4*x).*exp(x);
>> plot(x,f)
```

Grafici (cont.)

- Per creare grafici con colori e stili diversi:
  - Specificare dopo le coordinate una stringa di 2 caratteri:
  - Es:

```
>> t=0:0.01:2*pi;
>> y=sin(t);
>> plot(t,y,'g.')
>> axis equal
```



Possibili scelte

- L'insieme delle scelte possibili è il seguente:

r	red	.	point
g	green	o	circle
b	blue	x	x-mark
w	white	+	plus
m	magenta	*	star
c	cyan	-	solid
y	yellow	:	dotted
k	black	--	dashed
		-.	dash-dot

Grafici, comandi

- Altri comandi sono:

  `grid` : sovrappone al grafico un grigliato

  `title` : aggiunge un titolo del disegno

  `xlabel` : aggiunge una legenda per l'asse x

  `ylabel` : aggiunge una legenda per l'asse y

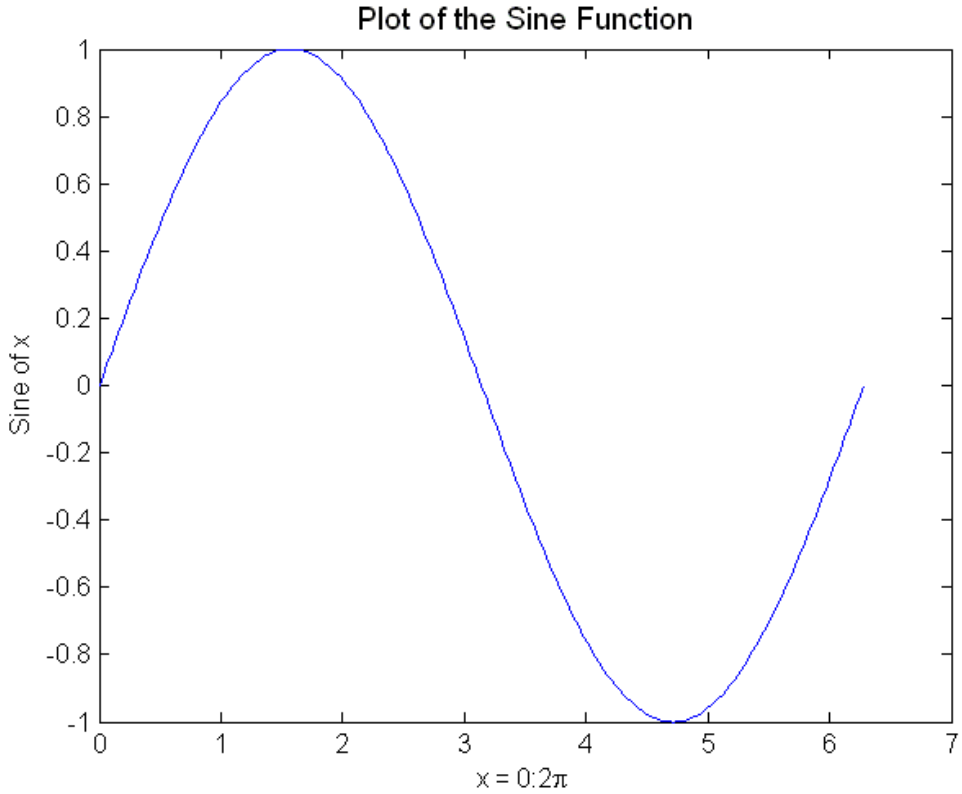
  `axis` : riscala gli assi del grafico

  `clf` : cancella il grafico corrente

- Il comando `figure` crea una nuova finestra grafica in cui far comparire il disegno; per spostarsi sulla n-esima finestra grafica, basta digitare `figure(n)`

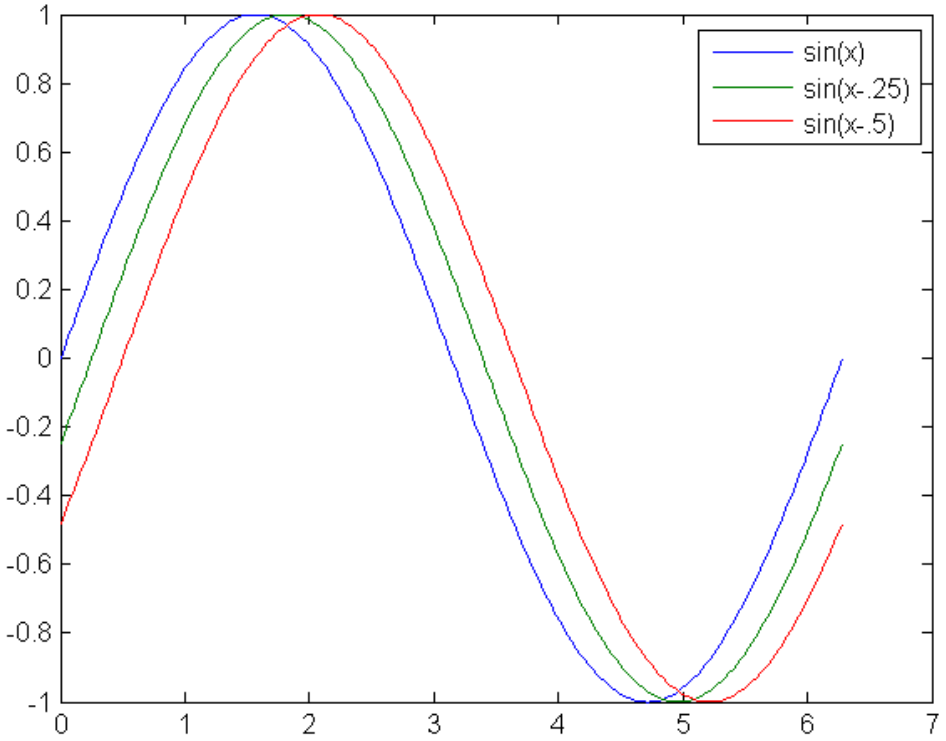
Esempio di grafico

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
xlabel('x = 0:2\pi')
ylabel('Sine of x')
title('Plot of the Sine Function','FontSize',12)
```



Grafici multipli

```
» y2 = sin(x-.25);
» y3 = sin(x-.5);
» plot(x,y,x,y2,x,y3)
» legend('sin(x)','sin(x-
  .25)','sin(x-.5)')
```



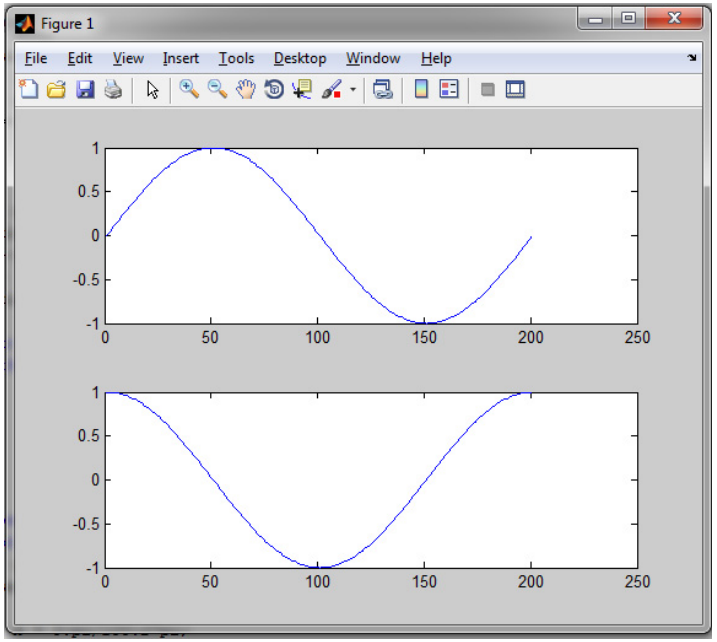
- Alternativa: hold on

Subplot

- **subplot**: Per visualizzare più grafici nella stessa finestra (non sovrapposti)
- 3 parametri: #parti verticali, #parti orizzontali, in quale parte eseguire il plot successivo

- Es:

```
x = 0:pi/100:2*pi;
y = sin(x);
z = cos(x);
subplot(211); plot(y);
subplot(212); plot(z);
```



Esercizio 1

- MATLAB does not have a function to determine which elements of an array are even numbers (i.e., . . . -4, -2, 0, 2, 4 , . . .). Write a function for this purpose, with the following specifications:

```
function E = iseven(A)
%ISEVEN Determines which elements of an array are even numbers.
%   E = ISEVEN(A) returns a logical array, E, of the same size as A,
%   with 1s (TRUE) in the locations corresponding to even numbers
%   and 0s (FALSE) elsewhere.
%   A must be a numeric array ( i.e. A = [-3, 2 , 5 , -1, 0, -2, 4] ).
```

Use of while or for loops is not allowed.
Hint: Become familiar with function floor.

Esercizio 2

Given

- an (m x n) matrix A
- and a (1 x n) vector v,

we want to subtract v from every row of A.

Es: A = 5 5 5
4 4 4

v = 1 2 3

ans = 4 3 2
3 2 1

Hint: you could use the function “repmat”

Esercizio 3

- given an mxn matrix A, create a matrix B of the same size of A, such that B contain the elements of A that are greater than zero, zero elsewhere.
- Es A = 2 -3 5
  B = 2 0 5

Esercizio 4

- Write an M-function with the following specifications:

- function H = imcircle(R, M, N)
- %IMCIRCLE Generates a circle inside a rectangle.
- % H = IMCIRCLE(R, M, N) generates a circle of radius R centered
- % on a rectangle of height M and width N. H is a binary image with
- % 1s on the circle and 0s elsewhere. R must be an integer >= 1.

- Your program must check the validity of R and also it should check to make sure that the specified circle fits in the given rectangle dimensions. Use of for or while loops is not permitted.
- *Hint:* Review function meshgrid and become familiar with function floor.

Esercizio 3

- The main purposes of this exercise are to learn how to work with displaying and changing directories and how to use that information to read an image from a specified directory. Write an M-function with the following specifications:

```
function [I, map] = imagein(path)
%IMAGEIN Read image in from current-working or specified directory.
%   I = IMAGEIN displays a window containing all the files in the
%   current directory, and saves in I the image selected from the
%   current directory.
%   [I, MAP] = IMAGEIN variable MAP is required to be an output
%   argument when the image being read is an indexed image.
%   [ . . .] = IMAGEIN('PATH') is used when the image to be read
%   resides in a specified directory. For example, the input
%   argument 'C:\MY_WORK\MY_IMAGES' opens a window showing
%   the contents of directory MY_IMAGES. An image selected from
%   that directory is read in as image I.
```

- *Hint:* Use online help to become familiar with functions cd, pwd, and uigetfile, imread.