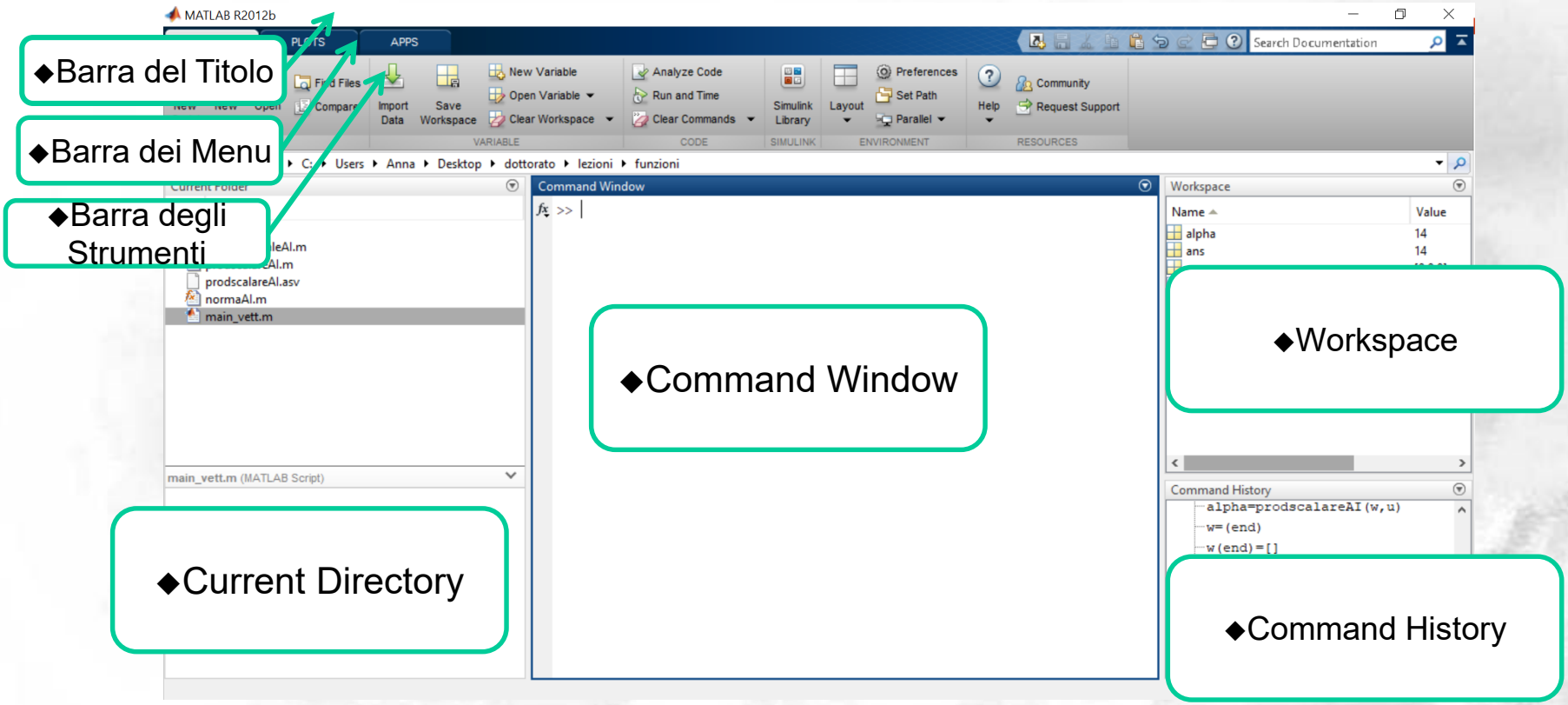
Introduzione a Matlab

Dott.ssa Anna Innac

- MATLAB (Matrix Laboratory) è un prodotto della Mathworks il cui sito web è: http://www.mathworks.com .
- MATLAB è un ambiente di sviluppo che integra calcolo matematico, grafica e programmazione scientifica il calcolo matriciale e la visualizzazione grafica, è possibile definire variabili, eseguire la valutazione di espressioni, valutare funzioni, eseguire grafici.
- MATLAB è, inoltre, dotato di applicazioni specifiche dette TOOLBOX.
- MATLAB è un linguaggio di programmazione di tipo interpretativo:

  un *programma* è una serie di istruzioni o comandi MATLAB contenuti in un file, detto *m-file*; ogni istruzione del programma è interpretata e, se priva di errori di sintassi, è eseguita immediatamente; mediante tale linguaggio è possibile creare nuove funzioni all'interno di MATLAB. Una *funzione* è un insieme di istruzioni contenute in un m-file, precedute da una istruzione speciale (*function*) che specifica i parametri della nuova funzione (ossia i dati che elabora e i risultati che ottiene).

Desktop di Matlab



VARIABILI

- Una *variabile* è un contenitore per un valore e corrisponde ad una locazione di memoria in cui è conservato quel valore; l'indirizzo di tale locazione è dato dal nome assegnato alla variabile.
- Ogni variabile ha un nome di fantasia, scelto dall'utente, lungo al più 31 caratteri (lettere, cifre, underscore), di cui il primo deve essere una lettera (*pippo, A1, var_dati,…)*.
- Un'*espressione* è costituita da costanti, variabili, operatori, funzioni predefinite e definite dall'utente (*ex_1=(a+b)/2, ex_2=b-a, ...*).
- Il valore dell'espressione assegnato ad una variabile specificata dall'utente **sovrascrive** il precedente contenuto se tale variabile esisteva già. Se l'utente non ha specificato variabili, il risultato dell'espressione è assegnato ad *ans*.

Operatori

Aritmetici

+ Somma

- Sottrazione

* Moltiplicazione

/ Divisione

^ Potenza

Relazionali

> maggiore

>= maggiore o uguale

== uguale

~ negazione

Logici

& and

| or

Xor or esclusivo

Altri

:

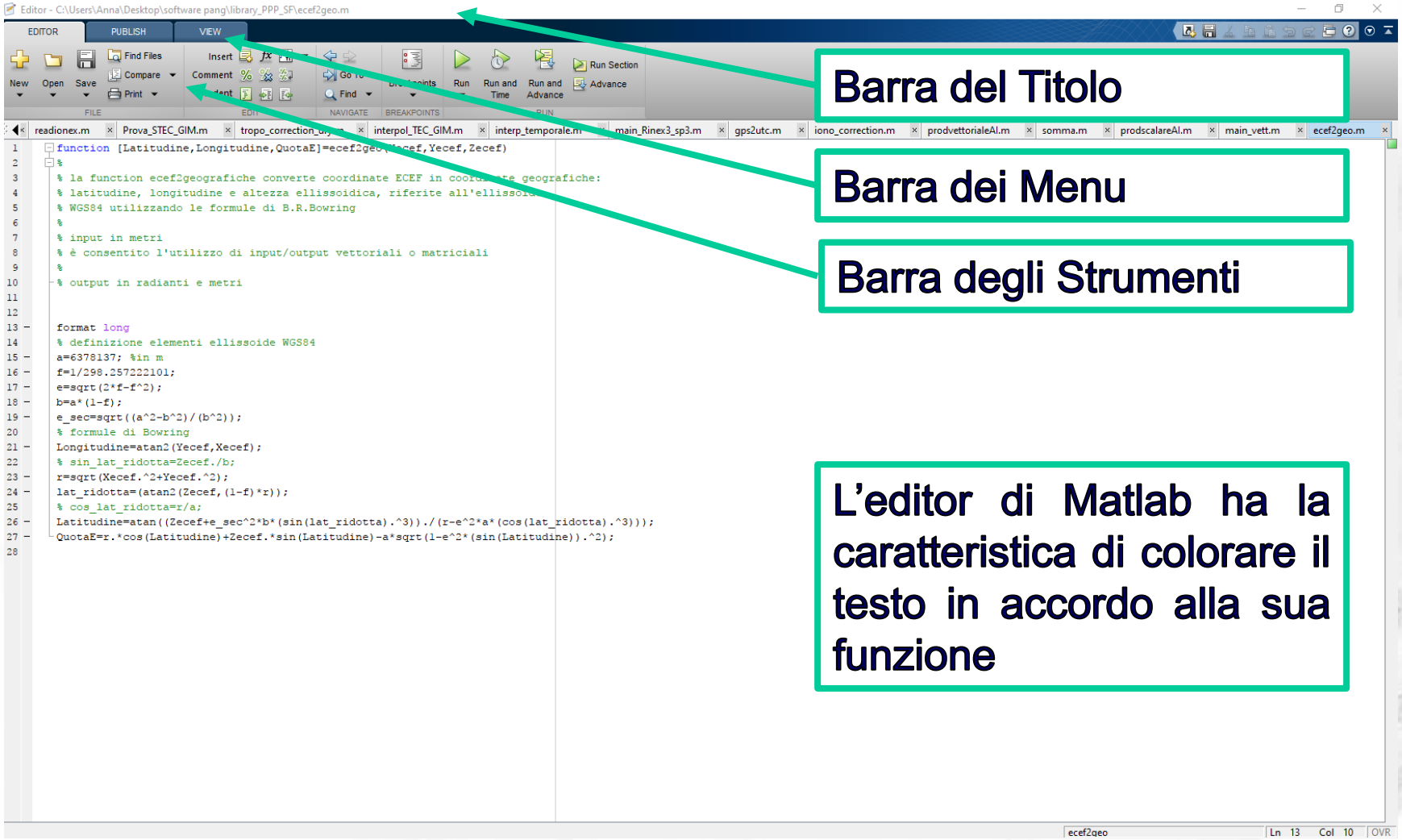
.

- == è un operatore di confronto
- = è un operatore di assegnamento

Gli Script M-file di MATLAB

- Matlab consente di memorizzare una sequenza di istruzioni in un file, detto *M-file*.
- Gli *script M-file* vengono creati, modificati e corretti mediante un *editor* all'interno dell'ambiente MATLAB, ma si possono selezionare anche programmi di editor diversi da quello associato a Matlab quali TEXTPAD.
- Per creare uno script *M-file* si può selezionare FILE/NEW/BLANK M-FILE o premere sul pulsante NUOVO della barra delle applicazioni. Si apre la finestra dell'*editor* in cui inserire le istruzioni del file, da salvare denominandolo con un nome di fantasia e estensione *.m*.
- Per editare un *M-file* già esistente, si può usare FILE/OPEN o premere il pulsante APRI della barra delle applicazioni. Viene aperta una finestra di dialogo, entro cui si individua quale file aprire. Il file compare nella finestra dell'editor e può essere cambiato e salvato.
- L'editor può essere attivato per modificare un M-file esistente anche mediante il comando
- *>>edit nome-file.m*

L'Editor di MATLAB



Barra del Titolo

Barra dei Menu

Barra degli Strumenti

L'editor di Matlab ha la caratteristica di colorare il testo in accordo alla sua funzione

Le Function

In Matlab è possibile definire delle proprie funzioni.

Il file contenente la function è a sua volta un m-file e deve essere salvato col nome della function stessa, per permettere a Matlab di trovarlo nel momento in cui è chiamato dal main.

Una function può ricevere e/o restituire un numero qualsiasi di variabili.

La sintassi per scrivere una function è:

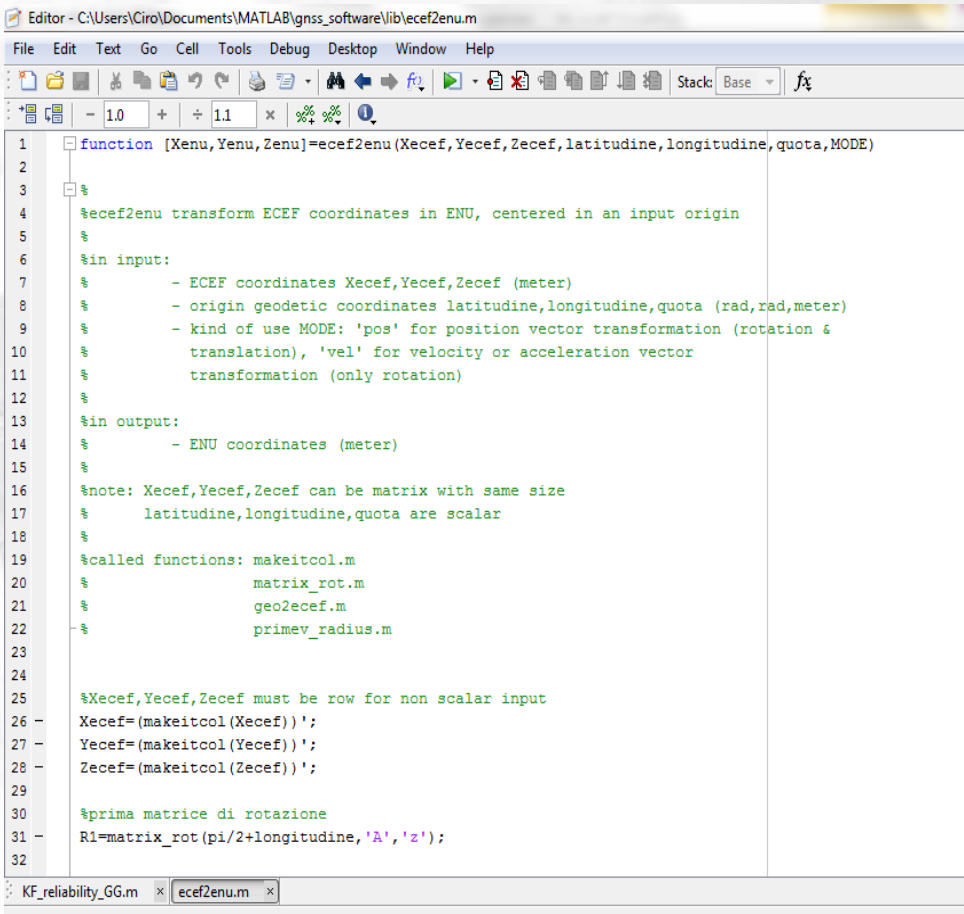
function [output]=nome_function(input)

La sintassi per richiamare una function è:

[output]=nome_function(input)

Input= una o più variabili in Ingresso

Output= una o più variabili che Saranno il risultato della funzione



Strutture di controllo

Le strutture di controllo sono costrutti sintattici per gestire il flusso di esecuzione di un programma, ovvero servono a specificare se, quando, in quale ordine e quante volte devono essere eseguite le istruzioni che lo compongono.

Le strutture di controllo tipiche sono di due tipi:

Alternative
- If

Iterative (cicli)
- For
- While

Il costrutto **if** valuta un'espressione logica ed esegue un gruppo di istruzioni quando l'espressione è vera

- I costrutti opzionali **else** ed **elseif** permettono l'esecuzione di gruppi di istruzioni alternative
- La parola **end** indica che l'ultimo gruppo di istruzioni è terminato

```
if espressione
        istruzioni
elseif espressione
        istruzioni
else
        istruzioni
end
```

ESEMPIO

◆Creare uno script che preveda l'inserimento a video di un valore n e che riconosca se n è pari o dispari

```
pariDispari
n= input( 'inserire il valore di n' )
if rem(n,2) == 0
    disp('Il numero è pari')
else
    disp('Il numero è dispari')
end
```

Strutture di controllo

La sintassi del ciclo **for** è semplicissima:

for indice=start:step:stop

...istruzioni...

end

In questo caso il contatore viene incrementato con un determinato passo (step) partendo da un valore (start) fino ad un valore fissato (stop).

Se non viene indicato espressamente lo step, l'incremento sarà unitario

Esercizi

Creare uno script che restituisca la somma di tutti i numeri che vanno da 1 a N (N inserito a video)

Soluzione ex. sommatoria:

```
N= input( ‘inserire il valore di N’ )
somma=0
for i=1:N
somma=somma+i;
end
```

Vettori

- Per introdurre un vettore riga è sufficiente riportare i valori separati da spazi bianchi o virgole

>>w=[1 2 3]

Oppure

>>w=[1, 2,3]

- Per introdurre un vettore colonna basta riportare fra parentesi quadre i valori delle componenti del vettore stesso separati da un punto e virgola

>>v=[1;2;3]

- Il comando v=[1:10] genera un vettore riga di dieci componenti dato dai valori 1,2,...,10.
- Il comando v=[1:0.5:10] genera un vettore riga di venti componenti dato dai valori 1,1.5,2,2.5,...,9.5,10, ovvero con passo 0.5.

- Per accedere alla componente di un vettore, ad esempio alla seconda

>>v(2)

- Attenzione: in Matlab l'indicizzazione inizia da 1 e non da zero!
- Esiste in Matlab la parola chiave **end** per accedere all'ultimo elemento di un vettore

>>v(end)

- Per controllare la dimensione di una variabile, usiamo il comando **size**

>>size(v)

- dato un vettore v, il comando **length(v)** ne restituisce la lunghezza.

manipolazione di sottoblocchi di vettori e di concatenazione

Siano v=[1 2 3 4 5] e w=[10 20].

- Per sostituire alle ultime due componenti di **v** le componenti di **w**, scriviamo

```
>> v=[1 2 3 4 5]; w=[10 20];
>> v(end-1:end)=w;
```

- Per eliminare da **v** la terza e la quarta componente usiamo il vettore vuoto []:

```
>> v=[1 2 3 4 5];
>> v(3:4)=[];
```

- Infine, per concatenare due vettori usiamo la sintassi

```
>> z=[v w]
```

Operazioni tra vettori

- Dato un vettore **v** di n componenti, si può calcolare in Matlab:
  - ➔ vettore trasposto: **v'**
  - ➔ Modulo di un vettore: norm(v)
- Siano ora **v**, **w** due vettori riga con componenti vi e wi, i = 1; …, n rispettivamente. Si ha:
  - ➔ somma algebrica **v** + **w** = $(v_1 + w_1, \ldots, v_n + w_n)$. In Matlab:

    >>v+w
  - ➔ differenza algebrica **v** - **w** = $(v_1 - w_1, \ldots, v_n - w_n)$. In Matlab:

    >>v-w

Operazioni tra vettori

- prodotto di uno scalare a per il vettore **v** genera un vettore dato da (av_1,….., av_n). In Matlab:

  >>a*v

- prodotto scalare (**v**,**w**) = (v_1w_1+…..+v_n w_n).
- prodotto componente per componente (attenzione: differente dal prodotto scalare!). Esso genera un vettore dato da (v_1w_1,….., v_n w_n). In Matlab:

  >>v.*w

- prodotto vettoriale (**v** x **w**)=[(v_2w_3-v_3w_2),(v_2w_1-v_1w_3),)(v_1w_2-v_2w_1)] [1]

Se i due vettori non hanno la stessa dimensione, si genera un errore

[1] Dati i due vettori dello spazio vettoriale di dimensione 3

Esercizi

- Generare un vettore riga **v** contenente gli interi da 28 a 80 con passo 10
- imporre 6° elemento =100
- togliere 4° elemento
- aggiungere in coda = [10 11 12]
- creare il vettore colonna di **v**
- creare un secondo vettore riga **z** (con interi da 1 a 8 e passo 1) e verificare le dimensioni dei due vettori
- Sommare i due vettori e calcolarne la norma

Soluzione

```
>> v=28:10:80
v =
  28   38   48   58   68   78
>> v(6)=100
v =
  28   38   48   58   68  100
>> v(4)=[]
v =
  28   38   48   68  100
>> v=[v 10 11 12]
v =
  28   38   48   68  100   10   11   12
>> z=1:8
z =
   1    2    3    4    5    6    7    8
>> v+z
ans =
  29   40   51   72  105   16   18   20
>> norm(ans)

ans =

   1.491006371549096e+02
```

Esercizi

- Creare una funzione che restituisca la norma di un vettore (a) dato in ingresso:

c=norma(a)

- Length(x)=Length of largest array dimension

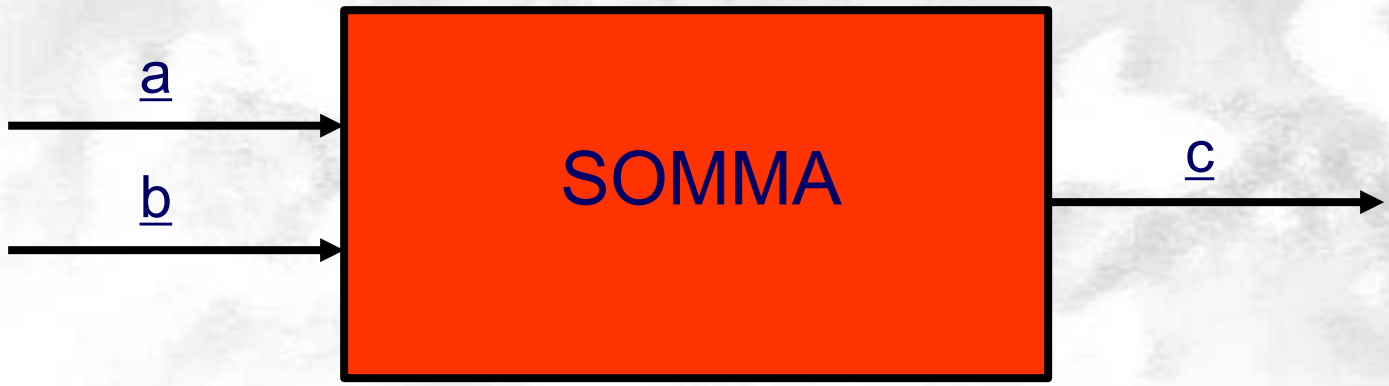
- function norma=norm_AI(a)

```
Dim_a=length(a);
somma=0;
for i=1:Dim_a
somma=(somma+a(i)^2);
end
norma=sqrt(somma);
```

```
norma=norm(a);
```

Esercizi

- Creare una funzione che dati due vettori in ingresso restituisca un output pari a:
  - ➔ Somma di due vettori
  - ➔ Prodotto scalare tra due vettori



c=somma(a,b)

SOMMA

```
function c=sommaAI(a,b)
n=length(a);
m=length(b);
   if n==m
    for i=1:n
        c(i)=a(i)+b(i);
    end
else
    disp('errore dimensione vettori non uguale')
end
```

PRODOTTO SCALARE

```
Function [scal]=prodotto_scal(a,b)
Dima=length(a);
Dimb=length(b);
scal=0;
if Dima==Dimb
For i=1:Dim
scal=scal+a(i)*b(i);
End
else disp('errore dimensione vettori')
end
```

```
scal=dot(a,b);
scal=a*b’
```

Esercitazione

Creare un main in cui vengono inseriti 2 vettori dati in ingresso a 3 diverse function che eseguono

- la norma di ogni vettore
- La somma dei due vettori
- La proiezione del primo sul secondo.

I vettori sono a=[18 -7 34] b=[-3 11 27]

Risultato (39.102; 29.309)

Risultato C=[15,4,61]

Risultato (787)

Editor - C:\Users\Anna\Desktop\lezioni\main_vett.m

EDITOR PUBLISH VIEW

New Open Save Find Files Compare Print — FILE

Insert Comment Indent — EDIT

Go To Breakpoints Run Run and Run and Run Section Advance

main_vett.m

```
1 -  a=[18 -7 34];
2 -  b=[-3 11 27];
3 -  norm_a=norm_AI(a);
4 -  norm_b=norm_AI(b);
5 -  scal=prod_scal_AI(a,b);
6
```

Editor - C:\Users\Anna\Desktop\dottorato\anna innac\esercizi matlab\normaAI.m*

EDITOR PUBLISH VIEW

New Open Save Find Files Compare Print — FILE

Insert Comment Indent — EDIT

Go To Find — NAVIGATE

Breakpoints — BREAKPOINTS

Run Run and Time Run and Advance Run Section Advance — RUN

main_vett.m normaAI.m*

```
1    function norma=normaAI(vettore)
2 -  n=length(vettore);
3 -  somma=0;
4 -  for i=1:n
5 -  somma=(somma+vettore(i)^2);
6 -  end
7 -  norma=sqrt(somma);
```

Esercizio

Creare un main in cui vengono inseriti 2 vettori tridimensionali dati in ingresso ad una function che restituisce un vettore ortogonale ai primi 2. Le uscite della function saranno il vettore, il suo modulo ed i versori

I vettori sono a=[-3 -7 24] b=[12 10 -27]

Risultato ([-51 207 54], 219.9227, [-0.2319 0.9412 0.2455])

vers(c)=c./norm(c)

[c, norma, vers]=prodvettoriale(a,b)

```
function [c, norma, vers]=prodvettoriale(a,b)
Dim_a=length(a);
Dim_b=length(b);
If Dim_a<=3
If Dim_a==Dim_b
    c(1)=a(2)*b(3)-a(3)*b(2);
    c(2)=a(3)*b(1)-a(1)*b(3);
    c(3)=a(1)*b(2)-a(2)*b(1);
    c=[c(1) c(2) c(3)];
else disp('errore')
end
norma=norma(c)
vers=c./norma
else
Disp('errore')
end
```

c=cross(a,b)

Matrici

- Per definire una matrice in MATLAB si elencano i suoi elementi per righe racchiudendoli tra parentesi quadre; gli elementi di ciascuna riga sono separati da almeno uno spazio o dalla virgola; le colonne sono separate premendo il tasto INVIO o usando il punto e virgola.
- **ESEMPIO.** Si consideri una matrice di *2x3* elementi, denominata M:

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

o in maniera equivalente

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

- Per specificare un elemento di un vettore occorre usare il nome del vettore, seguito tra parentesi tonde da un indice
- Per specificare un elemento di una matrice occorre usare il nome della matrice, seguito tra parentesi tonde da una coppia ordinata di indici separati da virgola che indicano la posizione di riga e quella di colonna
- M(i,j) i=numero riga; j=numero colonna
- **ESEMPIO**

  >>M(2,3)

  M(2,3)=

  6 (elemento di posizione *2,3* di *M*)

- Se le dimensioni delle matrici (o dei vettori) sono incompatibili per il tipo di operazione da eseguire viene visualizzato un messaggio di errore.
- Le operazioni operano elemento per elemento se uno degli operandi è scalari.

C=A+B	somma di matrici di egual dimensione *mxn*
C=A-B	differenza di matrici di egual dimensione *mxn*
C=A*B	prodotto di matrici A(*mxn*) e B(*nxp*)
C=A^p	elevamento a potenza solo con esponenti p interi positivi
C=A'	trasposizione di matrice
C=inv(A)	inversa di A
C=A\ B	risoluzione di sistemi: inv(A)*B
C=A/B	calcolo di A*inv(B)
C=A.*B	esegue A(i,j)*B(i,j)
C=A.\B	esegue B(i,j)/A(i,j)
C=A./B	esegue A(i,j)/B(i,j)
C=A.^B	esegue A(i,j)^B(i,j)
[m,n]=size(A)	dimensione per riga e per colonna di una matrice
l=length(u)	numero di elementi del vettore u
d=det(A)	determinante di una matrice quadrata

Creare una matrice A (5x4) e moltiplicarla per se stessa. (Verificare se dimensionalmente tale prodotto è possibile)

A*A'

Creare una matrice i cui elementi siano il quadrato della matrice A.

A.^2

- Dati due vettori v1 e v2 le cui componenti sono

v1= (5, 6, 3)

v2= (4 ,10, 12)

1) Generare i due vettori assegnandone i valori alle variabili v1 e v2;

2) Calcolare la matrice A uguale a v1 trasposto per v2;

3) Estrarre da A la sottomatrice B di dimensioni 2x2 fatta dalle ultime due righe di A e le ultime 2 colonne di A

```
>> v1=[5,6,3]
v1 =
     5     6     3
>> v2=[4 10 12]
v2 =
     4    10    12
>> A=v1'*v2
A =
    20    50    60
    24    60    72
    12    30    36
>> B=A(end-1:end, end-1:end)
B =
    60    72
    30    36
```

Creare le seguenti matrici:

A=[10 5 8 -20; -30 15 53 6; 12 4 3 -10; 47 53 7 98];
B=A*2.5;

- Visualizzare le due matrici appena costruite
- Estrarre da A la sottomatrice C di dimensioni 3x2 fatta dalle ultime tre righe di A e le prime due colonne di A;
- Calcolare la matrice D uguale alla matrice di A per l'inversa di A;
- Con l'operatore **find**, trovare gli indici degli elementi maggiori di 0.5 nella matrice A;
- Con l'operatore **find**, trovare gli indici degli elementi nell'intervallo [1, 10] nella matrice B;
- Con gli operatori **max** e **min** trovare i massimi e i minimi delle matrici A e B

```
>> A =
    10     5     8   -20
   -30    15    53     6
    12     4     3   -10
    47    53     7    98
>> B =
 25.0000   12.5000   20.0000  -50.0000
 -75.0000   37.5000  132.5000   15.0000
  30.0000   10.0000    7.5000  -25.0000
 117.5000  132.5000   17.5000  245.0000
>> C=A(end-2:end,1:2)
C =
   -30    15
    12     4
    47    53
```

```
>> D=A*inv(A);
>> [i,j]=find(A>0.5);
>> [i,j]=find(B>1 & B<10);
>> max(A)
ans =
 47    53    53    98
>> min(A)
ans =
 -30     4     3   -20
```

For matrices, max(X) is a row vector containing the maximum element from each column.

Grafici

La funzione PLOT ha forme diverse, dipendendo dagli argomenti di entrata.

- Se **v** è un vettore, plot(v) produce un grafico lineare degli elementi di v contro l'indice degli elementi di v.
- Se si specificano due vettori come argomenti, plot(x,y) produce un grafico di y contro x.

Per esempio, per diagrammare il valore della funzione seno da 0 a 2π, uso:

```
>>t = 0:pi/100:2*pi;
>>y = sin(t);
>>plot(t,y)
```

- Creare una seconda funzione di t e sovrapporre il grafico sul primo

```
>>y2 = sin(t - 0.40);
>>plot(t,y)
>>hold on
>>plot(t,y2)
```

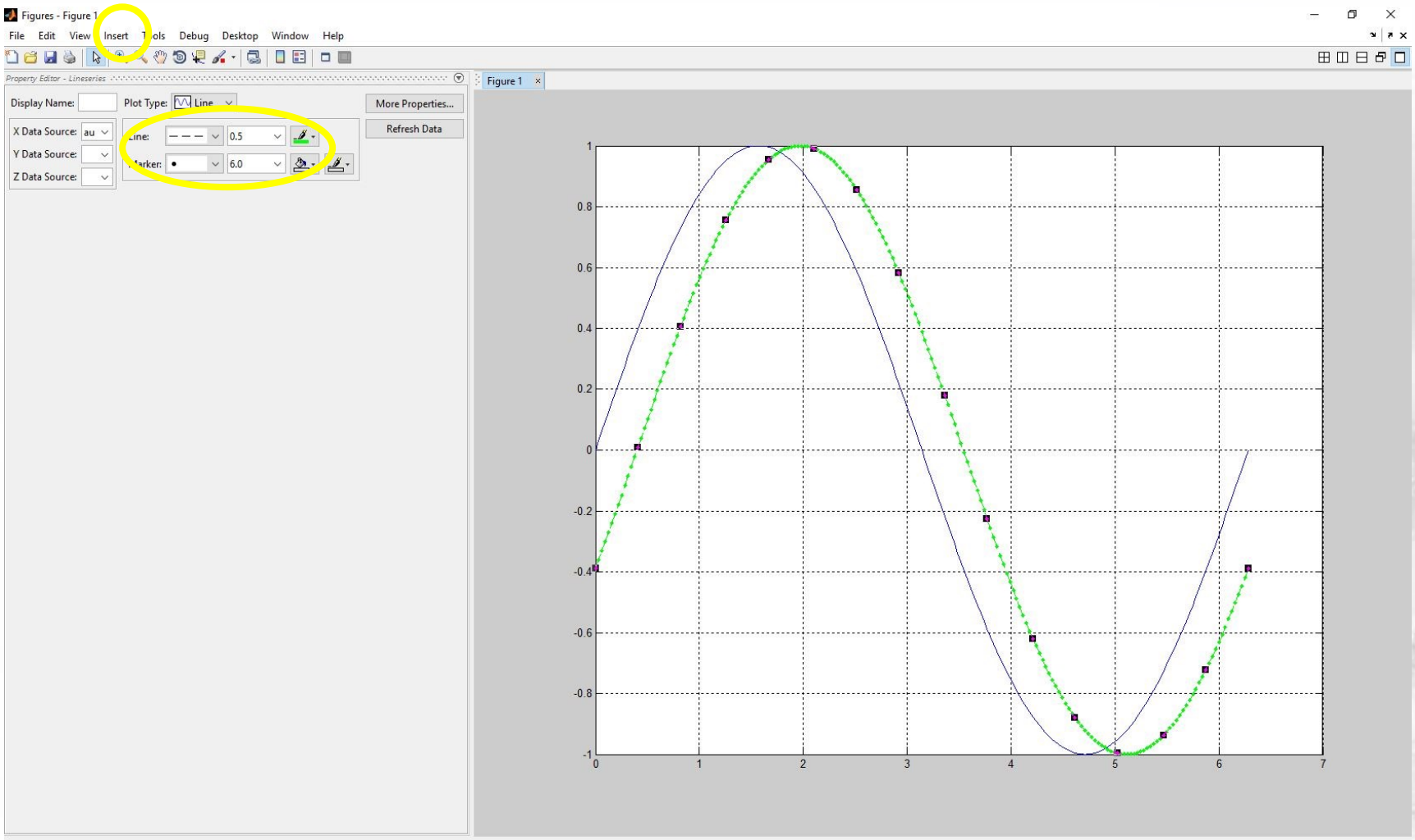
- È possibile specificare colore, stile della linea, e marcatori attraverso il seguente comando:

```
plot(x,y, 'marcatore-stile-colore')
```

Per esempio:

```
>>Figure
>>plot(t,y,'r--*')
```

help plot


Figures - Figure 1
File Edit View Insert Tools Debug Desktop Window Help
Property Editor - Lineseries
Display Name:
Plot Type: Line
More Properties...
Refresh Data
X Data Source: au
Y Data Source:
Z Data Source:
Line:
0.5
Marker:
6.0
Figure 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7