Calcolo Numerico

A.A. 2014-2015

Lab n. 1

15-10-2015

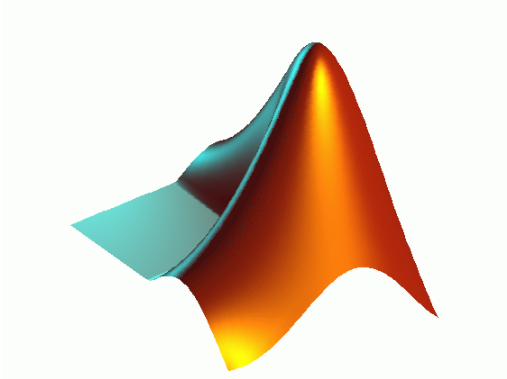
Info

Annalisa Pascarella

- email
  - **a.pascarella@iac.cnr.it**
- webpage
  - **http://www.iac.rm.cnr.it/~pasca**
- materiale del corso
  - slide delle esercitazioni, programmi MATLAB
- laboratorio? quanti siamo?

MATLAB/OCTAVE

- Introduzione a Matlab
- Rappresentazione dei numeri
- Underflow e overflow
- Vettori
- Cenni di programmazione
- Script
- Grafica 2D
- Cancellazione numerica
- Errori di arrotondamento



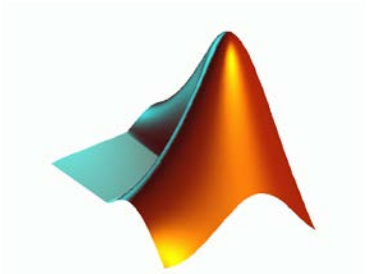
MATLAB

- **Matlab (MATrix LABoratory)** è un sistema software integrato per il calcolo scientifico sviluppato a partire dagli anni '70 utilizzabile sia in **maniera interattiva** che come **linguaggio di programmazione**.
- La struttura dati di base è la **matrice**, per la quale sono già predefinite **numerosi tipi elementari** (matrice identità, matrice nulla, matrice unità...), **funzioni algebriche e di manipolazione** (somma, prodotto, calcolo del determinante).
- Fornisce un ambiente di **calcolo**, **visualizzazione** e **programmazione scientifica** in cui è possibile:
  - calcolare direttamente espressioni matematiche
  - utilizzare il semplice ambiente di programmazione per creare i propri algoritmi
  - creare **grafici** in 2 e 3 dimensioni

Altre funzionalità di MATLAB

- Esistono vari pacchetti per i diversi tipi di applicazioni (**Toolbox**): l'elaborazione numerica dei segnali e delle immagini, la simulazione di sistemi dinamici, il calcolo simbolico, wavelet, ecc.
- Per informazioni su Matlab: **www.mathworks.com**
- Matlab è un software a pagamento. **Octave** è un software gratuito che ne riproduce buona parte delle funzioni fondamentali. Per info vedere

  https://www.gnu.org/software/octave/download.html



MATLAB linguaggio per programmare

- E' un **linguaggio interpretato e non compilato**: questo significa che le istruzioni vengono tradotte in linguaggio macchina (il linguaggio "capito" dal processore) e subito eseguite una per volta.
  - questo fatto si riflette in una **maggiore lentezza** di esecuzione rispetto ad un linguaggio compilato
  - Matlab possiede istruzioni molto potenti per la manipolazione di vettori e matrici. Se si riesce a fare ricorso il più possibile a queste istruzioni, evitando di utilizzare cicli for che, come vedremo, accedono singolarmente a ciascun elemento di un vettore o di una matrice, i tempi di esecuzione miglioreranno sensibilmente

- Utilizzando C, Fortran, C++, la traduzione da linguaggio ad alto livello a linguaggio macchina avviene invece nel **processo di compilazione**, in cui tutto il programma viene tradotto in linguaggio macchina e poi eseguito.

MATLAB

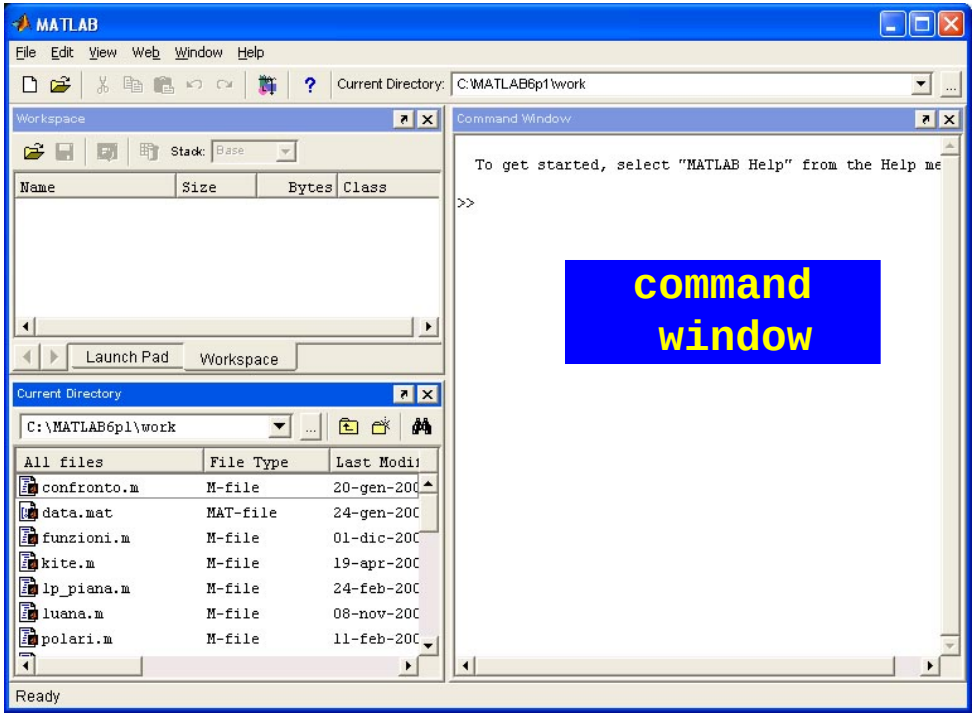
- Per lanciare Matlab da ambiente **Windows** basta cliccare con il mouse sull'icona corrispondente
- In ambiente **Unix**, digitare il comando matlab e quindi dare il comando di invio

- All'avvio viene aperta una finestra (spazio di lavoro) nella quale è possibile digitare comandi dal **prompt** **>>**

- Per entrare in confidenza con l'ambiente di lavoro è utile:
  - lanciare il comando **demo** che illustra le potenzialità del software attraverso significativi esempi numerici e casi test;
  - fare costante riferimento all'uso dell'**help**, ad esempio **`help sqrt`** (calcolo della radice quadrata di un numero).

MATLAB - Finestre

- **Command window:** finestra principale DI LAVORO INTERATTIVA. Contiene il **prompt >>**
- **Command history:** contiene tutti i comandi digitati da prompt per ripeterli, basta selezionare il comando di interesse con un click
- **Workspace:** contiene tutte le variabili usate e memorizzate automaticamente



MATLAB

Linguaggio di programmazione interpretato: legge un comando per volta eseguendolo immediatamente. Matlab lavora in **modo interattivo**

- l'utente digita una istruzione sul **prompt >>** ed ha immediatamente la risposta

```
>>  comando    (Per eseguire,  digitare Enter)
```

Esempio

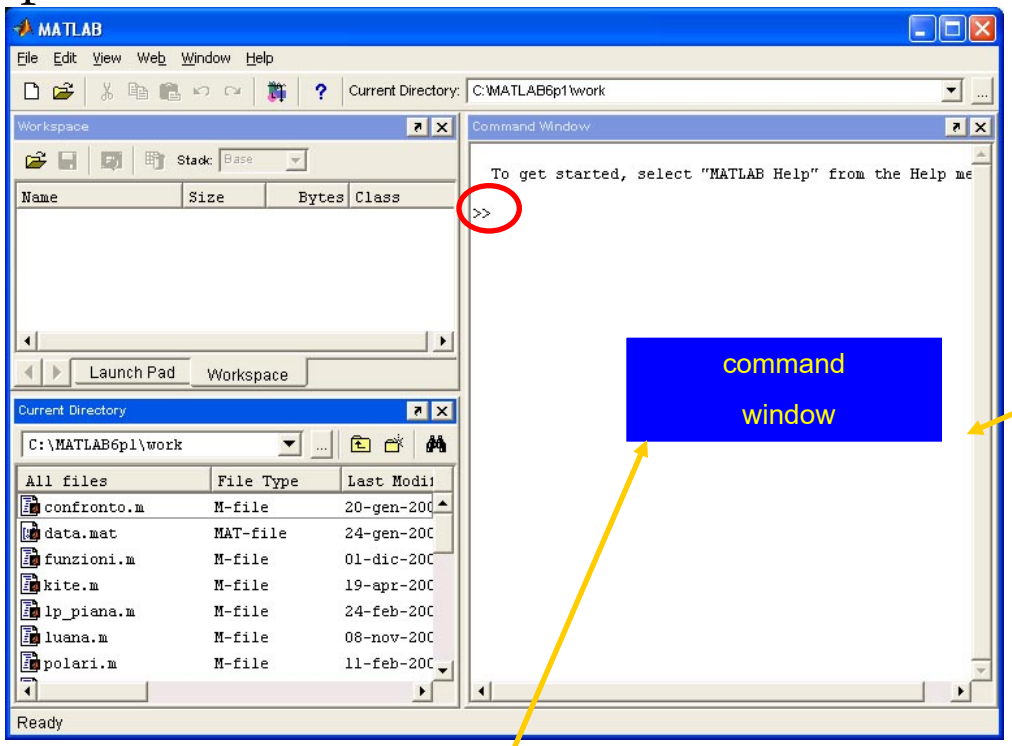
```
>> 3+2  (Enter)
ans =
     5
>>
```

Per uscire dalla sessione di lavoro interattiva:

```
>> quit
```

MATLAB come calcolatrice

- Può essere utilizzato in modo diretto per calcolare semplici espressioni matematiche:



- E' possibile definire **variabili** e operare su esse

```
x = 9 -> Enter
```

Variabili

- Una **variabile** è un insieme di dati modificabili
  - situati in una porzione di memoria destinata a contenere dei dati
  - suscettibili di modifica nel corso dell'esecuzione di un programma

- Ogni variabile è caratterizzata da un nome
  - una variabile è un contenitore, identificato da un nome univoco, di un qualsiasi valore, sia esso numero o stringa
  - anche in Matlab è possibile definire variabili
  - tipi di variabili
  - variabili di tipi diversi occupano spazi di memoria differenti

- Le variabili sono quindi definite da un **tipo** e da un **nome**

Variabile ans

- Se alle espressioni non si assegna una variabile la risposta è del tipo **ans=...,** dove **ans** è una variabile generata automaticamente da MATLAB.

```
>> 3+2

ans =

     5
```

- In Matlab non è necessario definire le variabili. Esse vengono automaticamante definite in seguito ad una assegnazione
  - La **assegnazione** è data dal comando =

Esempio

```
>> d = 2;
```

attribuisce alla variabile d il valore 2 (verificare nel **workspace**)

Variabili - definizione

Le variabili definite dall'utente possono essere utilizzate in altre espressioni matematiche

```
>> a = 8 + 3 - 1
a =
10

>> b = 1/sqrt(2)
b =
1.4142

>> c = a + b
c =
11.4142
```

Nota: c'è differenza tra maiuscole e minuscole

Variabili - visualizzazione

Per visualizzare il contenuto di una variabile, basta digitare il suo nome

Esempio: per visualizzare il contenuto di b

```
>> b
b =
     8
```

Oppure usare il comando **disp**

```
>> disp(b)
     8
```

- le variabili sono sovrascrivibili
- per cancellare una variabile si usa il comando **clear**
- le **variabili predefinite** possono essere ridefinite (esempio **i**)

Variabili - whos

Le variabili sono create automaticamente al momento del loro uso. Per conoscere le variabili create si può digitare il comando **whos** che da anche indicazioni utili sullo spazio occupato in memoria dalle variabili

```
>> whos
nome della variabile  dimensione   memoria occupata tipo
```

Esempio: se d è un numero intero

```
>> whos
  Name        Size                Bytes      Class

  d           1x1                     1      int8 array

Grand total is 1 elements using 1 bytes
```

Variabili

- E' importante sottolineare che, a differenza di altri linguaggi di programmazione, non è necessario specificare il tipo della variabile usata, ma questo risulta implicitamente definito in seguito all'assegnazione dei valori che deve assumere
- Tutte le variabili numeriche sono memorizzate in Matlab in formato **double**, ovvero in doppia precisione

- `clear nome_variabile`
  - per cancellare una variabile dal workspace
- `clear all`
  - per cancellare tutte le variabili dal workspace
- `clc`
  - pulisce lo schermo

Variabili

- Più espressioni o più istruzioni MATLAB possono convivere sulla stessa riga, basta che siano separate da un punto e virgola (se vogliamo sopprimere la visualizzazione di qualche risultato) oppure da una virgola (se vogliamo visualizzare il risultato)
  - per evitare la visualizzazione del risultato basta concludere l'espressione con un **punto e virgola**
  - per visualizzare il contenuto di una variabile è sufficiente scriverne i nomi eventualmente separati con una **virgola**

Esempio

```
>> a=2+3/5; b=sqrt(a), c=a*b
b =
1.6125
c =
4.1924
```

Diary

Per salvare la sessione di lavoro interattiva, o parte di essa, si usa il comando **diary**

```
>> diary on
>> diary nome.m
   ( sessione di lavoro che si vuole conservare)
>> diary off
   (parte che non si vuole conservare)
>> diary on
   (si vuole conservare di nuovo)
>> diary off
```

Conserva sia gli input che gli output ma anche tutti i messaggi di eventuali errori sul file di testo

Si può memorizzare come **Script** (vedremo in seguito) escludendo tutti gli output generati e gli eventuali messaggi di errore.

Save

- Per conservare il contenuto delle variabili si deve invece salvare tutta l'area di memoria (o parte di essa) con il comando **save**. Ad esempio:

```
>> x = 1;
>> a = 3;
>> z = sqrt(a)-x;
>> save areawork
>> save variables x z
```

I file prodotti (binari) hanno estensione **.mat**. Per ricaricarli nello spazio di lavoro usiamo il comando load:

```
>> load areawork
>> load variables
```

MATLAB come calcolatrice

Sono definite le **operazioni elementari**

- operazioni elementari **somma** + **differenza** -
  prodotto * **divisione** /
- operatori logici **and** & **or** | **not** ~
- operatori relazionali **maggiore** > **maggiore o uguale** >=
  minore < **minore o uguale** <=
  uguale == **diverso** ~=
- **elevamento a potenza** ^

MATLAB come calcolatrice

Costanti predefinite

Variabile	Significato
`ans`	valore ultima operazione eseguita non assegnata a variabile
`i,j`	unità immaginaria
`pi`	π, 3.14159265...
`eps`	precisione di macchina
`realmax`	massimo numero macchina positivo
`realmin`	minimo numero macchina positivo
`Inf`	∞, ossia un numero maggiore di `realmax`
`NaN`	Not a Number, tipicamente il risultato di un'espressione `0/0`

numero massimo rappresentabile (2^{1023}) **`realmax`**

numero minimo rappresentabile (2^{-1022}) **`realmin`**

Nonostante sia possibile ridefinire tali variabili è buona regola non farlo, eccezion fatta per le variabili i, j

MATLAB come calcolatrice

Funzioni predefinite Oltre alle operazioni di base, molte delle funzioni presenti in una calcolatrice scientifica sono presenti anche in MATLAB. Per una lista completa digitare **`help elfun`**

Funzione	Significato
sin	seno
cos	coseno
asin	arcoseno
acos	arcocoseno
tan	tangente
atan	arcotangente
exp	esponenziale
log	logaritmo naturale
sqrt	radice quadrata
abs	valore assoluto
sign	la funzione segno

Alcune funzioni predefinite in MATLAB

Una **funzione** necessita di alcuni parametri in ingresso, elencati tra parentesi tonde, e solitamente restituisce un risultato che può essere assegnato ad una variabile.
Per esempio l'espressione

```
>> y=cos(pi/4)
y =
0.7071
```

utilizza la funzione coseno con argomento **`pi/4`** e ne assegna il risultato alla variabile **`y`**.

Help!

Per informazioni sulle funzioni di Matlab (vedere anche l' help da menù)

```
>>   help nome_funzione
```

informazioni su una specifica funzione

Esempio: come si usa la funzione log?

```
>> help log

 LOG    Natural logarithm.
    LOG(X) is the natural logarithm of the elements of
  X.
    Complex results are produced if X is not positive.

    See also LOG2, LOG10, EXP, LOGM.
```

Help!

Digitando solo il comando **help** si ha l' elenco degli argomenti (pacchetti disponibili)

```
>> help

HELP topics:

matlab\general     -  General purpose commands.
matlab\ops         -  Operators and special characters.
matlab\lang        -  Programming language constructs.
matlab\elmat       -  Elementary matrices and matrix manipulation.
matlab\elfun       -  Elementary math functions.
matlab\specfun     -  Specialized math functions.
matlab\matfun      -  Matrix functions - numerical linear algebra.
matlab\datafun     -  Data analysis and Fourier transforms.
matlab\audio       -  Audio support.
```

>> help nome_argomento

Produce l'elenco e la descrizione delle funzioni relative all'argomento selezionato

Lookfor

Se non si conosce il nome delle funzione, si usa il comando **lookfor**

```
>> lookfor  parola chiave
```

Produce l'elenco e la descrizione di tutte le funzioni che sono in qualche modo legate alla parola chiave.

Esempio: esiste una funzione che produca la matrice identità di una certa dimensione?

```
>> lookfor 'identity matrix‘

EYE Identity matrix.
SPEYE  Sparse identity matrix.
```

A questo punto si chiede l' **help** della funzione che interessa

Format

- Il risultato dell'operazione **`cos(pi/4)`** è visualizzato con quattro cifre decimali, questa è l'impostazione standard di Matlab

- In Matlab tutte le variabili sono in doppia precisione (forma a virgola mobile, **floating point**), ovvero sono rappresentate internamente con 64 bit, cui corrispondono 15 cifre significative decimali.

- Tutti i calcoli vengono effettuati in doppia precisione, mentre diversa è la visualizzazione delle variabili che viene determinata con il comando format.

- Il formato esterno può essere deciso dall'utente: (**`help format`**)

- Tale comando serve per modificare il formato di visualizzazione dei risultati ma NON la precisione con cui i calcoli vengono condotti.

Format

```
>>  format  short   %  4  cifre  significative  dopo  la  virgola
   (opzione di default)
>> sqrt(2)
ans =
    1.4142

>> format short e  % forma esponenziale (potenze di 10)
>> exp(10)
ans =
  2.2026e+004

>> format long       % 14 cifre dopo la  virgola
>> sqrt(2)
ans =
   1.41421356237310

>> format long e          % forma esponenziale
>> exp(10)
ans =
    2.20264579480672e+004
```

Precisione macchina

- Eps prende il nome di **precisione macchina** ed è legata all'approssimazione ottenibile con l'insieme dei numeri macchina a disposizione
  - rappresenta quella costante caratteristica di ogni aritmetica floating-point ed è la massima precisione con cui vengono effettuati i calcoli sul calcolatore; è il piu piccolo numero sentito dall'aritmetica dei numeri macchina
- **eps** è il minimo valore tale che **(1+eps) > 1**
  - è il più numero che sommato a 1 da un numero maggiore di 1
- Sul calcolatore che stiamo usando

```
>> eps
?
>> 1+eps/2 -1
0
```

Classi di dati

- **double:** numeri in doppia precisione compresi tra -10^{308} e 10^{308} (8 bytes per elemento)
- **uint8**: interi a 8 bits per elemento senza segno compresi tra 0 e 255 (usato per le immagini)
- **uint16**: interi a 16 bits per elemento senza segno compresi tra 0 e 65535
- **uint32**: interi a 32 bits per elemento senza segno compresi tra 0 e 4294967295
- **int8**: interi a 8 bits per elemento con segno compresi tra –128 e 127
- **int16**: interi a 16 bits per elemento con segno compresi tra -32768 e 32767
- **int32**: interi a 16 bits per elemento con segno compresi tra
- -2147483648 e 2147483647
- **single**: numeri in singola precisione compresi tra -10^{38} e 10^{38} (4 bytes per elemento)
- **char**: caratteri (2 bytes per elemento)
- **logical**: 0 o 1 (1 byte per elemento)

Classi di dati

I nomi delle classi sono anche funzioni che permettono la conversione da una classe ad un'altra

Esempio: se x è una variabile double

```
>> int8(x)
```

converte x in una variabile intera

I caratteri char si indicano tra 2 apici

Esempio: attribuire alla variabile A il carattere f

```
>> A = 'f';

>> disp(A)
f
```

Trucchetti...

- Durante la sessione di lavoro è possibile richiamare i comandi precedentemente digitati utilizzando il tasto ←

- Immettendo i primi caratteri di un'istruzione già digitata e poi premendo il tasto ⟷ viene completata la riga con l'ultima istruzione che inizia con quegli stessi caratteri;

- Con il tasto sinistro del mouse sulla finestra di calcolo si possono selezionare parti di testo che è poi possibile copiare, tagliare ed incollare sulla linea di comando.

Numeri complessi

- I numeri in virgola mobile (la rappresentazione che un calcolatore fa dei numeri reali) non sono l'unico tipo di dato numerico ammesso. Un altro tipo di dato utile in varie applicazioni e presente in MATLAB sono i **numeri complessi** e le operazioni con questi.
- Un numero complesso z, in forma algebrica **`z = Re(z) + iIm(z)`** (con **`Re(z), Im(z)`** parte reale e parte immaginaria), può essere scritto in modo simile anche in MATLAB:

```
>> a=3+4i;
```

- L'utilizzo di operazioni su numeri complessi è ammesso

```
>> a=3+2i;
>> b=3.6+2.4*i;
>> a+b
ans =
6.6000 + 4.4000i
```

Esercizi

- Assegnare alla variabile **a** il valore **4+2log(π/2)/5**
- Calcolare in **b** il valore **e^cos(2.4)**
- Calcolare in **c** il valore **b/4**
- Visualizzare **a, b, c** in formato corto esponenziale
- Visualizzare gli stessi valori in formato long. Poi tornare al formato di default
- Se **x=5, y=3, z=2** calcolare **(3x-4)²/(5y -z)**
- Calcolare $y = \dfrac{e^{x+1} - \sqrt{(x+1)^3}}{x^3 + \ln x + 1}$ per **x=10**
- **Che output producono le seguenti istruzioni?**
  - **x=1.e-15;((1+x)-1)/x**
  - **x=0;sin(x)/x**

Cancellazione numerica

- La cancellazione numerica è la perdita di cifre significative
- E' un fenomeno che si verifica durante l'operazione di **sottrazione** tra due numeri "quasi uguali"
  - se due numeri sono quasi uguali, dove uguali s'intende a meno della precisione macchina, allora è possibile il verificarsi della cancellazione numerica.
- Siano *x1 e x2* due numeri reali. Se *x=x1-x2* è "molto piccolo", l'errore relativo

$$\delta_x = \left| \frac{fl(x1 - x2) - x}{x} \right|$$

può essere molto grande e ciò produce una perdita di cifre significative nel calcolo di *fl(x1-x2)*

- E' sempre preferibile evitare la sottrazione tra numeri macchina "quasi uguali"

Esercizio

- Calcolare numericamente le soluzioni dell'equazione di secondo grado

$$ax^2 + bx + c = 0$$

con le seguenti formule

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \qquad x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \qquad x_2 = \frac{2ac}{-b - \sqrt{\Delta}}$$

per i valori di

  - *a=1, b=206.5, c=0.01021*
  - *a=1, b=50000, c=0.01*

- Calcolare il valore dell'equazione per tali valori e il prodotto *x1x2*

Vettori: definizioni e operazioni

Lavorare con MATLAB

- In MATLAB tutte le variabili sono trattate come **matrici** (non a caso l'acronimo MATLAB sta per **MATrix LABoratory**). Anche gli scalari sono visti come una matrice.

scalari	->	matrici 1 x 1
vettori riga	->	matrici 1 x n
vettori colonna	->	matrici n x 1
matrici	->	matrici m x n

$\mathbf{v} = (v_1, \ldots, v_n)$

$\mathbf{v} = (v_1, \ldots, v_n)^T$

$$A = \begin{pmatrix} a_{11} \ldots a_{1n} \\ \vdots \\ a_{m1} \ldots a_{mn} \end{pmatrix}$$

Vettori riga

- In Matlab si possono definire facilmente vettori e matrici
- Un **vettore** si definisce elencando le sue componenti separate da uno spazio e racchiudendole tra parentesi quadre [ ]

- **Vettore riga**

```
>>  x = [10 20 30 40]
x =
     10     20     30     40
```

è equivalente a

```
>>  x = [10,20,30,40]
x =
     10     20     30     40
```

  - in questo caso le componenti sono separate da una virgola

Vettori colonna

- **Vettore colonna**

```
>> x=[10; 20; 30; 40]
x =
      10
      20
      30
      40
```

  - anche per visualizzare il contenuto di variabili che sono vettori si può usare il comando **disp**

```
>> disp(x)
     10
     20
     30
     40
```

Vettori

- Per convertire un vettore riga in uno colonna (e viceversa) si usa il comando ' (**apice**) che produce il trasposto della variabile a cui è applicato

```
>>  v=x'

v =

     10     20     30     40
```

- Per estrarre un elemento di un vettore:

```
nome_vettore(posizione elemento)
```

Esempio: estrarre il secondo elemento di v

```
>> v(2)

ans =

     20
```

Nota: Gli indici di un vettore sono sempre numeri interi e strettamente positivi e la numerazione parte da 1!

Generazione di vettori con :

Il comando **:** (colon) può essere usato per **generare vettori**

```
Nome_vettore = minimo:incremento:massimo
```

Esempio: Generare un vettore costituito da elementi compresi tra 1 e 5 con incremento 1

```
>> x = 1:1:5

x =

     1     2     3     4     5
```

Quando l'incremento è uguale a 1 (**passo di default**) è possibile ometterlo

```
>> x = 1:5

x =

     1     2     3     4     5
```

Generazione di vettori con :

Esempio: Generare un vettore costituito da elementi compresi tra 1.5 e 2 con incremento 0.1

```
>> x=[1.5:0.1:2]
x =
    1.5   1.6   1.7   1.8   1.9   2.0
```

Se si vuole una successione decrescente si deve specificare un **incremento negativo**

Esempio: Generare un vettore costituito da elementi compresi tra 100 e 80 con incremento -5

```
>> x=[100:-5:80]
x =
    100    95    90    85    80
```

Generazione di vettori con :

Esempio:

```
>> x=[3:0]
non produce niente!
```

- se non specificato, l'incremento è da intendersi pari a 1

```
>> x=3:-1:1
x =
     3 2 1
```

Comando linspace

- Per generare **vettori equispaziati** contenuti in un certo intervallo si può usare anche il comando **`linspace`**
  - può essere molto utile nel caso si consideri un passo che non sia intero
  - al comando devono essere forniti come parametri di ingresso i due estremi dell'intervallo e il numero di elementi del vettore N (per default è 100). Restituisce un vettore di lunghezza N il cui i-esimo elemento è dato da

```
Nome_vettore=linspace(minimo, massimo, N)
```

```
x(i) = minimo + (i-1)*(massimo-minimo)/(N-1)
```

Esempio: Generare un vettore costituito da 10 elementi compresi tra 1.5 e 2.4

```
>> x=linspace(1.5,2.4,10)

x =

    1.5    1.6    1.7    1.8    1.9    2.0    2.1    2.2    2.3
  2.4
```

Il comando :

Per **estrarre** contemporaneamente più di un elemento consecutivo di un vettore si usa il comando **:** (colon)

`nome_vettore(inizio:fine)`

Esempio: estrarre dal primo al terzo elemento di **`v=[10:10:40]`**

```
>> v(1:3)

ans =

     10     20     30
```

Esempio: estrarre dal terzo al quarto elemento di v

```
>> v(3:4)

ans =

     30     40
```

Il comando :

Esempio: estrarre tutti gli elementi di v

```
>> v(1:end)
ans =
     10     20     30     40
```

Oppure, se lo si vuole come vettore colonna,

```
>> v(:)
ans =
     10
     20
     30
     40
```

Il comando :

Per estrarre contemporaneamente più di un elemento di un vettore non consecutivi ed equispaziati

```
nome_vettore(inizio:passo:fine)
```

Esempio: estrarre gli elementi di v di indice pari (passo = 2)

```
>> v(2:2:end)
ans =
     20     40
```

Esempio: estrarre tutti gli elementi di v di indice pari ma da destra verso sinistra (passo = -2)

```
>> v(end:-2:1)
ans =
     40     20
```

Vettori

- Un vettore può essere usato per estrarre elementi non consecutivi e non equispaziati di un altro vettore

```
nome_vettore([pos1 pos2 pos3 …])
```

Esempio: Sia `v = [7 1 3 7 0 8 3],` estrarre gli elementi di v di indici 1 3 e 6

```
>> ind = [1 3 6];
>> v(ind)
ans =
     7     3     8
```

- In un'**unica istruzione**

```
>> v([1 3 6])
ans =
     7     3     8
```

Elimare elementi

- Un' operazione che può risultare utile è quella di **eliminare** alcuni elementi in un vettore cambiandone allo stesso tempo la dimensione

```
>> x = 1:10;
>> x(1:3)=[]
x =
4 5 6 7 8 9 10
```

Vettori

- Se un vettore (o una qualsiasi istruzione) è troppo lungo, prima di andare a capo vanno aggiunti 3 punti **…**

```
>> x = [3 1 6 7 9 10 4 29 6 0 …
           4 5 8 2 4 ]
x =
    3   1   6   7   9   10   4   29   6   0   4   5   8   2   4
```

- Se un elemento di un vettore è una espressione, non bisogna lasciare spazi all'interno dell'elemento, oppure l'espressione va racchiusa tra parentesi tonde

```
>> x = [1  6  3*2+1  4]
x =
     1     6     7     4
```

Oppure

```
>> x = [1  6  (3*2+1)  4]
x =
```

Vettori

I vettori non vengono dimensionati. **La loro dimensione può essere modificata in corso di lavoro**

Esempio: Sia **`x = [3 1 4 5]`** e si assegni il valore 10 all'ottavo elemento di x

```
>> x =  [3 1 4 5]

x =

     3     1     4     5

>> x(8) = 10

x =

     3     1     4     5     0     0     0    10
```

alle posizioni non definite viene assegnato il valore zero

Vettori

Esempio: Sia `x = [3 1 4 5]` e si elimini l'elemento in posizione 3

```
>> x(3) =  []
x =
     3     1     5
```

[] indica il **vettore vuoto**

Per conoscere la lunghezza di un vettore si usa il comando `length(x)`

Esempio: determinare la lunghezza del vettore x sopra definito

```
>> length(x)
ans =
     3
```

isempty

- Per controllare se un vettore è vuoto c'è la funzione **`isempty()`**
  - ritorna una variabile booleana

```
>> x(3) =  []
x =
     3     1     5
>> isemty(x)
ans =
     1
```

Operazioni

In MATLAB sono definite le operazioni dell'algebra lineare numerica di **moltiplicazione per uno scalare** e di **somma** e **sottrazione** tra vettori. Tali operazioni agiscono **componente per componente** e restituiscono un vettore della stessa lunghezza

```
>> x = 1:5;
>> y = [50 10 30 40 20];
>> 2*x
ans =
2 4 6 8 10
>> 2./x
ans =
2.0000 1.0000 0.6667 0.5000 0.4000
>> x+y
ans =
51 12 33 44 25
>> y-x
ans =
49 8 27 36 15
```

```
>> x(1:3)+y
??? Error using ==> plus
Matrix dimensions must agree
```

Operazioni puntuali

MATLAB estende le proprietà delle operazioni somma e sottrazione anche ad altre operazioni, fra cui **moltiplicazione** e **elevamento a potenza**. Il vincolo è che i due vettori operandi abbiano lo **stesso numero di componenti**

```
>> a = 1:3;
>> b = a;
>> a.*b
ans =
1 4 9
>> a.^b
ans =
1 4 27
```

Se le dimensioni non sono compatibili:

```
>> c= [1 2];
>> a.*c
Matrix dimensions must agree
```

Operazioni puntuali

Le operazioni precedenti (moltiplicazione puntuale, la divisione puntuale e l'elevamento a potenza puntuale) sono tipiche dell'ambiente MATLAB

- non hanno un corrispondente dal punto di vista dell'algebra lineare in quanto agiscono su vettori e matrici intesi come **strutture di dati** più che entità matematiche.
- L'istruzione `x.*y` utilizza la moltiplicazione puntuale tra vettori e fornisce un vettore con la proprietà che ogni sua componente è uguale al prodotto delle corrispondenti componenti dei vettori **x** e **y**.
- le stesse operazioni possono essere applicate nel caso di vettori colonna o più in generale nel caso di **matrici**. La cosa essenziale è che gli operandi siano dello stesso tipo ed abbiano le stesse dimensioni.

Operazioni

Uniche eccezioni a questa regola sono date dal caso in cui le precedenti operazioni vengano applicate tra un **vettore** ed una **costante**. In tal caso MATLAB considererà la costante come un vettore di pari dimensioni avente tutte componenti costanti. Ad esempio

```
>> x = 1:5;
>> x+1
ans =
2 3 4 5 6

>> 1-x
ans =
0 -1 -2 -3 -4

>> x.^2
1 4 9 16 25
```

Vettorizzazione e operazioni puntuali

- Molte funzioni predefinite in MATLAB accettano come argomenti dei vettori
  - questa caratteristica di MATLAB è molto importante in quanto consente di scrivere in forma molto chiara e compatta sequenze di istruzioni eliminando in molti casi l'uso di strutture e cicli che agiscono elemento per elemento

```
>> v = [4 9 16]
>> radici = sqrt(v)
   radici =
   2     3     4
```

Vettorizzazione e operazioni puntuali

Esempio: per costruire una semplice tabella di valori della funzione coseno nell'intervallo [0, π] possiamo procedere nel seguente modo

```
>> n = 5;
>> x = linspace(0,pi,n);
>> c = cos(x);
```

L'istruzione **c=cos(x)** applicata ad un vettore x restituisce un vettore c di uguali dimensioni e tipo con la proprietà che l'elemento di indice i è **c(i) = cos(x(i)).** Risulta quindi equivalente all'istruzione

```
>> c = cos(x);
>> c=[cos(x(1)) cos(x(2)) cos(x(3)) cos(x(4) cos(x(5))]
```

Funzioni

- Ecco alcune **funzioni MATLAB** che consentono di costruire particolari matrici e vettori.
  - queste funzioni MATLAB possono essere utilizzate con un diverso numero di parametri.
  - si consulti l'help per una descrizione dettagliata

Funzione	Significato
`linspace`	vettore riga di elementi equispaziati
`logspace`	vettore riga di elementi equispaziati in scala logaritmica
`zeros`	matrice contenente solo elementi uguali a zero
`ones`	matrice contenente solo elementi uguali a uno
`rand`	matrice contenente numeri casuali
`eye`	matrice identità
`diag`	matrice diagonale
`magic`	matrice a valori interi con somme uguali su righe e colonne

Funzioni

Esempio

```
>> zeros(1,3)
   ans =
   0     0     0

>> ones(4,1)
   ans =
   1
   1
   1
   1
```

Funzioni

```
length(v) max(v) min(v) sum(v) norm(v) abs(v) sort(v)
find(v > k) isempty(v)
```

Si consulti l'help per una descrizione dettagliata

max e min

- **max (min)** restituisce il valore massimo (minimo) contenuto nel vettore. Se si usa la funzione con **due parametri di output** il primo valore è il valore massimo, il secondo l'indice del vettore per cui si ha il valore massimo

Esempio

```
>> v = [2 4 1 6 9 3];
>> max(v)
   ans =
    9
>> [massimo i_massimo] = max(v)
   massimo =
    9
    i_massimo =
    5
>> disp(v(i_massimo))
    9
```

Esercizio

- Creare un **vettore** **`x`** che ha componenti con valori compresi tra 0 e 20, estremi inclusi, con incremento costante pari a 0.2.
  - determinare la lunghezza e memorizzarla in una variabile **`len`** di tipo **int**
  - **estrarre** gli elementi di **indice dispari** e assegnarli alla variabile **`y`**
  - **estrarre** gli elementi di **indice pari** di **`y`** procedendo da destra verso sinistra e assegnarli alla variabile **`z`**
  - **eliminare** il primo elemento di **`z`**
  - **assegnare** il valore –9 al terzo elemento di **`z`**.
  - sostituire il secondo elemento di z con $2\sin^2(\pi/4)\cos(\pi/4)$
  - **creare e visualizzare** il vettore **`w`** costituito da tutti gli elementi di **`z`** seguiti dai primi 2 elementi di **`z`** e gli ultimi 3 di **`z`**
  - calcolare il valore massimo e minimo di **`w`**
  - trovare gli elementi del vettore **`w`** maggiori di 5
  - utilizzare la funzione **`sort`** (cosa fa?) su **`w`**

Esercizi

- Assegnare alla variabile **x** il vettore costituito dai primi 20 numeri naturali. Estrarne il quarto elemento e moltiplicarlo per il quindicesimo
- Costruire il vettore v di 40 elementi **`v = [1,2,…,20,20,19,…,1]`**
- Creare un vettore **x** che ha 6 componenti con valori compresi tra 0 e 10, estremi inclusi, con incremento costante.
- Assegnati i vettori **`u = [1; 0; 2; -3] e v = [3; 0; 2; 1]`**
  - calcolarne il prodotto scalare; cosa fornisce invece il prodotto **`v*u`**?
  - calcolare i vettori colonna **`z;w;y`** definiti, componente per componente, da

```
zi = ui*vi; wi = ui^vi; yi = zi=wi
```