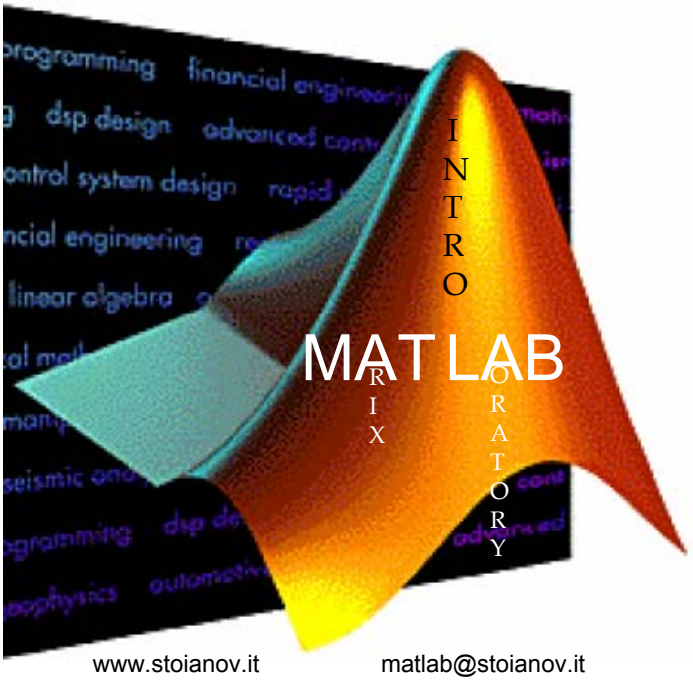


www.stoianov.it matlab@stoianov.it

24 ore

MATLAB in 12 ore …

- Informatica & Matlab: intro, IDE, variabili, script, flusso
- Codifica dei dati: matrici/strutture
- Interfaccia (in/out): grafici, file

PROGETTI, altro

- Un progetto guidato
- Opzionale: Matlab Open Source (Octave)
- Un progetto personale

Ref: ftp://ftp.dic.units.it/pub/Science/dispensa-Matlab.pdf

Programmi

- **Algoritmo**: **una seria di elaborazioni di dati** che:
  - hanno la finalità di realizzare un **determinato compito**,
  - partendo da un certo **stato iniziale** (contesto).
- **Dati**: **informazioni in forma digitale**; elementari ed/o strutturati. I dati strutturati semplificano l'elaborazione.
- **Programma**: **l'implementazione di un *algoritmo***
  - utilizzando una certa *struttura* di dati
  - in un particolare *linguaggio* di programmazione,
  - utilizzando gli elementi di base di questo linguaggio (***istruzioni***) e seguendo il ***sintassi*** di questo linguaggio, ossia le regole con quale si combinano gli istruzioni.

Programmazione

- **Identificare** (formalizzare) il **problema**:
  evento (descrizione) => elaborazioni => risultato
  dati *d'ingresso* => elaborazioni => dati *d'uscita*;
- Progettare la rappresentazione interna (struttura) dei **dati**.
- Disegnare un **algoritmo** che implementa le elaborazioni dei dati necessarie per raggiungere l'obbiettivo.
- Scegliere un **linguaggio** di programmazione.
- Disegnare l'**interfaccia** (grafica) con gli utenti (**GUI**).
- Scrivere un **programma** che implementa: l'algoritmo; l'archiviazione dei dati; l'interfaccia.
- **Compilare** il programma.
- **Verificare** il funzionamento del programma (*Debug*).

Far girare il CPU

- Il CPU capisce un limitato insieme di istruzioni codificati con numeri (ad es., "15" ≡ "*SOMMA*").
- Scrivere un modesta programma in questo codice (Dati, Elaborazione, GUI, File) richiede troppo tempo.
- Per poter realizzare programmi complessi, si usano linguaggi ad alto livello, ad es., C++, che permettano l'utilizzo di strutturi dei dati e del codice, è semplificano l'archiviazioni dei dati e l'implementazione di GUI.

Linguaggi di programmazione

Interpretati: ciascuna istruzione viene codificata al momento del esecuzione

Compilati: l'intera programma viene codificata in codice del processore.

		Algol		
		Pascal	**Matlab**	
	Fortran	C++	Lisp	Spreadsheet
				Database
Assembler	C	Java	Prolog	…

Complessità del linguaggio: la complessità delle strutture di codice e dati.

Ambiente Integrato di Sviluppo (IDE)

- **Compilatore / Interprete**: traduce i programmi in codice che il CPU è in gradi di capire
- **Editore** del programma (editore di testo) che:
  - Aiuta scrivere i programmi seguendo un certo stille
  - Per semplificare la lettura di un programma, codifica con diversi colori i vari tipi di elementi (istruzioni, variabili, operatori, …).
  - Interfaccia il compilatore (esegua la compilazione ed interpreta gli errori.
  - Possibilità di fare ***debug*****:** verificare il coretto funzionamento di ogni passo del programma.
- **Editore dell'interfaccia grafica** con l'utente (GUI)
  - Assomiglia un editore d'immagini, permette l'inserimento di vari elementi della interfaccia (bottoni, grafici, ecc);
  - Da accesso al codice di programma che elabora gli azioni associati
- **HELP:** provvede un accesso immediato ad ogni particolarità del linguaggio di programmazione (critico per i linguaggi "ricchi")

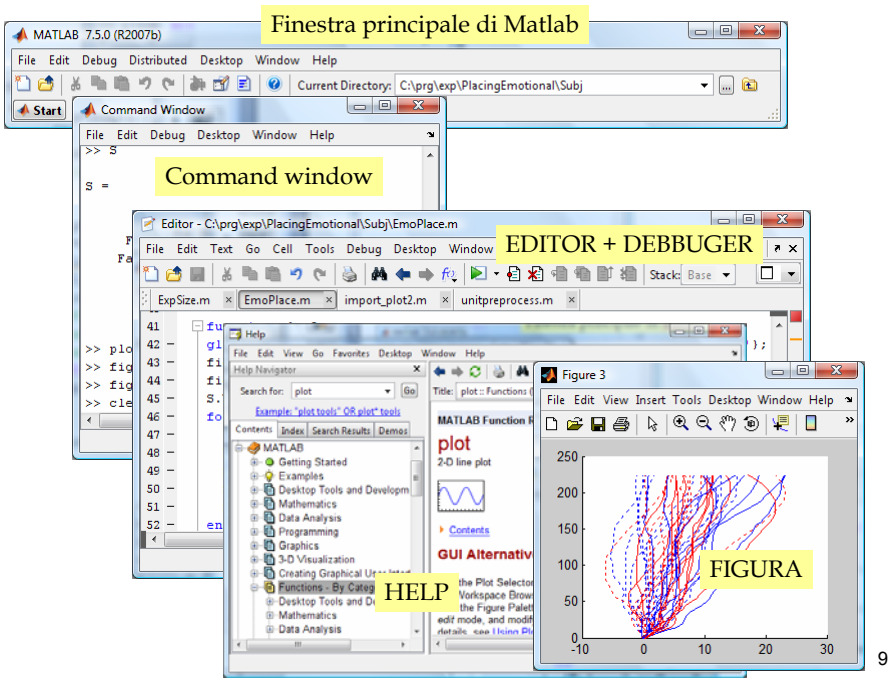
Matlab: linguaggio & ambiente di programmazione

- Intuitivo e potente **linguaggio** di programmazione
- Ambiente integrato per lo sviluppo di programmi (**IDE**)
- Ambiente di **interpretazione** dei programmi che integra:

  Dati: importazione, esportazione, gestione interna
  Elaborazione: calcoli complessi in poche righe
  Interfaccia: grafica (GUI) e testuale; grafici ad ogni tipo

  Toolbox: una ricca biblioteca di calcoli

- **Octave**: la versione "open source" di Matlab; compatibile per 99%. Manca un potente IDE. Minimo consumo di memoria. Meglio per alcuni aspetti; Peggio per altri.



9

Command (Consola)

- Consola: interfaccia testuale all'interpretatore di MATLAB ed al contenuto della memoria (WORKSPACE)
- Prompt: il simbolo « >> » all'inizio delle righe. Segnala che l'interpretatore è in attesa di una commanda.

  >>
- Un commanda inserita viene **esseguita immediatamente** e viene **visualizzato il risultato** (o errore, in rosso ☹)

```
>> 13*9
ans =
     117

>> sin(pi/6)
ans =
    0.5000
```

- I tasti ↑ e ↓ recuperano gli recenti comandi eseguiti.

Facciamo una prova: Calcolare il numero di ore in un anno solare …

Operazioni ed Espressioni

Operazioni: gli istruzioni del "linguaggio dei calcoli"

Aritmetici

```
+ addizione
- sottrazione
* moltiplicazione
/ divisione
^ potenza
```

Logici

```
&  and logico
|  or logico
~  not logico
```

Predicati

```
== uguale
~= diverso
<  minore
<= minore o uguale
>  maggiore
>= maggiore o uguale
```

Espressioni: sequenze di *operazioni* e *operandi* (numeri compresi), seguendo i "standard" matematici, ad. es., *(5+3)*4*

Variabili

- **Contenitori di dati (possono avere qualunque valore!!)**
- ad ogni livello di complessità (numeri, matrici, strutture)
- riferiti per nome* "i", "StrLen", "SubjName", …
- Assegnare un valore con l'istruzione "=" prima di usarli
  (in altri linguaggi si definisce anche il tipo di dati che contengono)
- Assegnare

```
>> i = 5
i =
     5
```

- Riferire:

```
>> i = i + 1
i =
     6
```

- *Workspace*: il contenitore di variabili in Matlab
  - Lettere minuscoli e maiuscoli sono diverse
  - Il *Workspace* per se è una variabile ... complessa

Esercizio: Assegnare 0 ad una variabile; Poi, moltiplicarla per 100. Il risultato ?

Inoltre,

- Espressioni lunghe si possono scrivere su più righe finalizzandole con " …"

```
>> A = (6+4)/2 – 7 + 9
>> A = (6+4)/2 …
   -7+9
```

- Testi (stringhe di lettere) si racchiudono in apostrofi ' '

```
>> 'movimento braccio contralaterale'
```

- Commenti: righe che cominciano con il simbolo % non vengono interpretati. Sono colorati in verde.

```
>> % Incrementare i tramite l'espressione i=i+1
```

- Non si visualizza il rizultato di espressioni seguiti da « ; »

```
>> 13*9;
```


Esercizio: Assegnare ad una variabile un testo che contiene il tuo nome.

Info

- Le variabili nel Workspace
  - who, whos elenca le variabili nel *workspace*
  - clear all rimuove tutte le variabili nel *workspace*
  - clear x,y,z rimuove le variabili indicati
- Cartella
  - pwd mostra l'attuale cartella (directory) di lavoro
  - cd nome cambia la cartella di lavoro
  - dir mostra il contenuto della cartella

Esercizio: (1) Quali variabili hai utilizzato fin ora? (2) Vedi la documentazione di "who"

M-script

- Invece di re-scrivere una sequenza di comandi, è possibile: (i) salvarla in un file con estensione ".m "

  File: MyFirstScript.m

- ed (ii) eseguirla digitando il nome del file* o con il tasto *F5*

  >> MyFirstScript

- Così, lo stesso script si può re-utilizzare in diverso contesto (cioè, diversi valori delle variabili utilizzati nello script), o modificando variabili-"parametri" al interno del script.

* Per eseguire un m-file, verifica che il file è presente nella cartella attuale.

Esercizio: Calcolare la superficie totale di 3 dischi con radio 1, 3, 9

Strutturare il programma

- L'implementazione di un algoritmo e la sua comprensione si facilità se il codice è **frazionato** in blocchi che …
- eseguano sotto-compiti **incapsulati**, indipendenti uno d'altro. I blocchi comunicano un tramite **interfaccia**: dedicati variabili d'*ingresso* e d'*uscita*.
- Questo stille sta nella base del *Object-Oriented Programming*, "**OOP**" dove sono "incapsulati" interi ***oggetti***; un oggetto comprende descrizioni (dati) e comportamenti (funzioni)
- In Matlab, questa frazione si realizza attraverso "*script*", e "funzioni". Una funzione ad alto livello esegue ("chiama") altri funzioni/script, dandone dati e raccogliendo i risultati.
- E' indispensabile descrivere il più possibile con **commenti** tutto ciò si esegue in qualunque script o funzione.

Esercizio: Frazionare l m'script dell'esercizio precedente. Mettere commenti.

Funzioni

- Problema: in un programma complesso, un script potrebbe per sbaglio modificare variabili utilizzati in un altro script (e.g., "i")
- Funzioni: "script" chiamati per nome, che utilizzano *variabili incapsulati*, cioè, definiti al interno della funzione e non visibili fuori dalla funzione se non dichiarati come *variabili d'uscita* o *global*. Una funzione può anche avere *variabili d'ingresso*.
- L'eccezione: i variabili ***GLOBAL*** (tutti i variabili nei script sono *global*):
  - risiedono nel "*workspace*"
  - sono visibili nelle funzioni dove sono dichiarati *global*
  - loro modifiche restano permanenti.

```
function [y1, y2] = <nome della funzione*>(x1, x2, …)
  global Z;
  …
  y1 = … ;
  y2 = … ;
```

* Il nome della funzione è meglio che coincide con il nome del file in cui risiede.

Esempio: Funzioni

Compito: 1) Implementare un calcolatore che fa la somma di due numeri con una funzione di nome **somma2**
2) Verificare il suo funzionamento nella consola.

File `somma2.m`

```
function s = somma2(i,j)
s = i + j;
```

CONSOLA

```
>> somma2(3, 10)
ans
     13
>> i=5;
>> j=1;
>> k=somma2(j,i)
ans
    6
```

Esercizio: Funzioni

Compito: Calcolare la superficie totale di 3 dischi, di radius 1.5, 2.0, 1.2, utilizzando una funzione che calcola la superficie di un disco.

File `discsurf.m`

```
function s = discsurf(r)
s = pi * r * r;
```

File `tredischi.m`

```
STot = discsurf (1.5);
STot = STot + discsurf (2.0);
STot = STot + discsurf (1.2)
```

CONSOLA

```
>>tredishci
```

CONSOLA

```
>>STot = discsurf (1.5)+ discsurf(2) + discsurf(1.2)
```


Interazione con l'utente

- x = input ('invito');

  chiede con un certo 'invito' nel *command* l' inserimento di un valore, che viene assegnato alla variabile x.

- disp(x);

  visualizza nel *command* il valore di una variabile

- fprintf('formato',var1,var2,…);

  visualizzazione formattata nel *command*

  fprintf('R=%.2f S=%.2f \n', R , S);

  fprintf('%d forme di tipo %s con radius %.2f \n', N, T, R ,);

% <type>
d integer
f float
c char
s string

Esercizio: Interfaccia

Compito: Calcolare la superficie totale di 2 dischi, di radius richiesto dal utente.

Utilizzare:

R=input (‘Radius of disk1 ?’);

```
fprintf(‘Surface=: %.1f’ \n’,S);
```

File `due_dischi.m`

```
R = input (‘Radius of disk1 ?’);
S = discsurf (R);
R = input (‘Radius of disk2 ?’);
S = S + discsurf (R);
fprintf(‘Total surface: %.1f \n’,S);
```

CONSOLA

```
>>due_dischi
```