UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
INDUSTRIALE

DIPARTIMENTO
MATEMATICA
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

MATLAB STEP BY STEP

Materiale realizzato da Michela Redivo Zaglia

con il contributo di E. Bachini, L. Bruni, W. Erb, A. Larese, F. Piazzon

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

DIPARTIMENTO MATEMATICA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Laboratorio di Calcolo Numerico LAB 1

Introduzione a Matlab e primi Esperimenti

Docenti: E. Bachini, L. Bruni

Email: elena.bachini@unipd.it Email: bruni@math.unipd.it

6 marzo 2024

Conoscenze preliminari

Per poter seguire con facilità questo tutorial è necessario:

- Aver realizzato l'Onramp Course della Matlab Academy
  `https://matlabacademy.mathworks.com/`

Il nostro compagno di viaggio

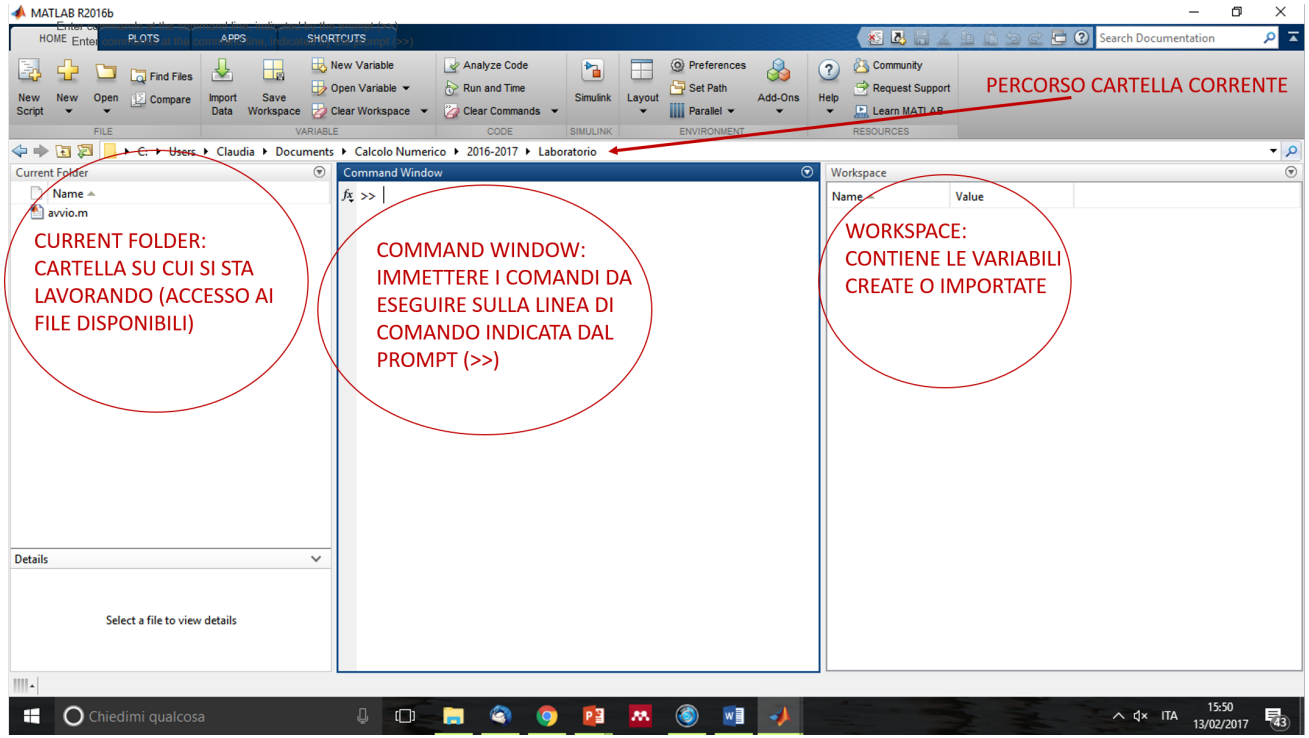
Matlab = **Mat**rix **Lab**oratory è un ambiente di programmazione (software commerciale)

- calcolatrice ad alta precisione
- in grado di operare (efficentemente e in modo estremamente facile per l' utente) su matrici e vettori
- programmabile (= possibili complicati calcoli)
- linguaggio di programmazione funzionale:
  - l'accento è posto sulle funzioni
  - la memoria è gestita in modo automatico
- linguaggio interpretato: i programmi che scriveremo non hanno bisogno di compilazione

MATLAB - Avvio

Aprire il programma MATLAB:

- tramite icona del programma
- oppure, da terminale o shell: digitare il comando `matlab`



Files, Cartelle e Schermate

- Matlab prevede di lavorare in una cartella di lavoro e "non sa" cosa c'è fuori
- Pannelli/schermate:
  - Command Window: ambiente che esegue operazioni "istantanee", comandi e programmi matlab
  - Workspace: elenco delle variabili in memoria
  - Current Folder: cartella di lavoro attuale e suo contenuto
  - Editor: è un editor di testo con suggerimenti
  - Help: digitando `help <nomecomando>` aiuto sintetico
  - Documentation: `doc <nomecomando>` documentazione completa
- Matlab legge molti tipi di files l'estensione più importante è ".m"
  - script: (li vedremo oggi) serie di comandi matlab ad esempio creazione di variabili e operazioni sui contenuti del workspace.
  - function: (le incontreremo nelle prossime lezioni) algoritmi di calcolo (e non solo) da poter eseguire all'occorrenza su dati variabili (input) che modificano solo i dati in uscita.

Creare variabili

- Una variabile è un contenitore con nome dove si possono memorizzare i dati voluti (o risultanti da un operazione)
- La creazione della variabile avviene (in genere) con un comando di assegnazione (operatore di assegnazione "="). Es:

```
>> A=5.3

A =

    5.3000

>>
```

- si può anche creare una variabile vuota

```
>> A=[]

A =

     []

>>
```

Creare variabili

N.B. cosa succede nel workspace?

Cosa succede se alla fine del comando aggiungo ";"?

```
>> A= 5.3;

>>
```

Sopprime la visualizzazione dell'assegnazione effettuata.
Però potete controllare nel workspace che la variable è stata creata correttamente.

Nomi di variabili

- Consigliato utilizzare solo caratteri alfanumerici, il primo deve essere obbligatoriamente alfabetico.
  N.B. ammesso il simbolo di trattino basso ma non il trattino intermedio (denota sottrazione!)
- Nella definizione e nell'uso, i nomi di variabili sono case sensitive.
- Non si possono inserire spazi bianchi all'interno di un nome di variabile!
  E non sempre viene segnalato errore: si provi con

```
>> var 1+2*var2-5.3
>> var1+2*var 2-5.3
>> var1 + 2 * var2 - 5.3
```

Variabili predefinite

- Esistono variabili predefinite (nomi variabile riservati)

```
pi      3.14159265...
eps     Floating-point relative precision,
realmin Smallest floating-point number,
realmax Largest floating-point number,
Inf     Infinity
NaN     Not-a-number
```

- E la variabile speciale

  `ans`

  creata automaticamente per il risultato di un operazione.

- Attenzione (**fortemente sconsigliato**): il valore di tali variabili può essere sovrascritto dall'utente. Es:

```
>> pi=1234

pi =

        1234
```

Tipi di variabili

- Non tutti i contenitori sono adatti ad un certo tipo di dati (`class` in matlab). Matlab in genere sceglie tipologia di contenitore in base al contenuto (no dichiarazione).
- Principali classi:

```
double          - double precision float.
single          - single precision float.
logical         - boolean.
```

- Altre:

```
uint8           - unsigned 8-bit integer.
uint64          - unsigned 64-bit integer.
int8            - unsigned 8-bit integer.
int64           - signed 64-bit integer.
```

- il nome-tipo-variabile è anche il comando di conversione...

Importanza del tipo I

```
>> a=1; class(a)

ans =

    'double'

>> b=int8(a); c=b/pi

c =

  int8

   0
```

Importanza del tipo II

```
>> mystr='questa e'' una variabile stringa';
>> mystr

mystr =

    'questa é una variabile stringa'

>> mynumstr=double(mystr)

mynumstr =

   113   117   101   115   116    97    32   101    39    32   117   110
          97    32   118    97   114   105    97

    98   105   108   101    32   115   116   114   105   110   103    97

>> char(mynumstr)

ans =

    'questa é una variabile stringa'

>>
```

pulizia workspace e command window

- `clear a`
  `clear('a')`
  Sono sintassi equivalenti che eliminano la variabile a dal workspace.
- `clearvars`
  Elimina invece tutte la variabili dal workspace.
- `clear all`
  Elimina tutto (variabili, functions,. . .)

E' buona norma in un programma eliminare tutte le variabili all'inizio del programma per non lavorare su dati erroneamente presenti nel workspace.

- `clc` pulisce la command window.

Operatori di base

- = assegnazione (da non confondere con) == operatore logico uguale (output 0/1)
- + addizione
- - sottrazione
- * moltiplicazione di due scalari o scalare-vettore, ma anche
- * moltiplicazione riga per colonna nelle matrici se compatibili
- / divisione di due scalari o vettore-scalare
- : "da a con passo 1"
- `x0:dx:x1` "da x0 a x1 con passo dx"
- ; "non mostra output del comando"

Esempi di built-in functions:

- `sum()` somma degli elementi di un vettore
- `prod()` prodotto
- `max()`, `min()` massimo e minimo di vettore

Help e documentazione

E' buona prassi utilizzare in modo intensivo `help` e `doc` per l'autocorrezione e per l'apprendimento.

- Supponiamo di conoscere il nome del comando (funzione) Matlab o della variabile riservata Matlab su cui vogliamo acquisire informazioni dettagliate.

  ```
  >> help log
  >> doc log
  ```

- Per una finestra di ricerca interattiva usare
  ```
  >> doc
  ```
- Per cercare la funzione che contiene nella sua descrizione "logarithm" si usa

  ```
  >> lookfor logarithm
  ```

I numeri in Matlab

- Matlab utilizza di default una **rappresentazione** numerica in doppia precisione, cioè a 64 bit, seguendo lo standard IEEE. Questo permette di rappresentare numeri macchina in valore assoluto nell'intervallo [realmin, realmax] con errore relativo di assegnazione $\left|\frac{\mathtt{fl}(x)-x}{x}\right|$ al più eps

```
>> eps
ans =
    2.2204e-16
>>
```

- Matlab usa per un numero la **notazione** decimale, con un opzionale punto decimale, un segno (+ o -) che lo precede e la notazione esponenziale come potenza di 10. Es:

3	-99	0.0001
9.6397238	1.60210e-20	-6.02252e23

Precisione numerica in Matlab

Matlab di default lavora in doppia precisione, ma possiamo decidere se memorizzare i numeri in singola (4 bytes) o doppia (8 bytes) precisione usando il comando `single` o `double`.

```
>> ad  = 1.23456789011223344556677889 9
ad =
     1.234567890112233e+00
>> as = single(ad)
as =
  single
   1.2345679e+00
```

Nota.

Attenzione, se si passa da precisione singola a doppia le cifre che Matlab aggiunge sono random.

```
>> ad2 = double(as)
ad2 =
     1.234567880630493e+00
```

Arrotondamento numerico in Matlab

Esistono dei comandi Matlab che permettono di effettuare l'arrotondamento di un risultato secondo diversi criteri:

`fix`	arrotondamento verso 0
`round`	arrotondamento verso l'intero più vicino
`floor`	arrotondamento verso $-\infty$
`ceil`	arrotondamento verso $+\infty$

```
>> a=3.7;
>> round(a)
ans =
     4
>> fix(a)
ans =
     3
```

```
>> ceil(a)
ans =
     4
>> floor(a)
ans =
     3
```

Formato di visualizzazione in Matlab

Matlab usa diversi formati di visualizzazione dei numeri che si possono modificare attraverso il comando `format`
Provate a digitare nella command windows

```
>> help format
```

per vedere tutti i formati disponibili.

Nota.

Il settaggio di `format` *NON influisce sulla precisione numerica.*

Il cambio di formato non influisce su come vengono realizzate le operazioni, specifica solamente come vengono visualizzati i numeri.

Formato di visualizzazione in Matlab

```
>> format long e
>> 0.1
ans =
     1.000000000000000e-01
>> format short e
>> 0.1
ans =
   1.0000e-01
>> format long
>> 0.1
ans =
   0.100000000000000
>> format short
>> 0.1
ans =
    0.1000
```

Il formato di default in Matlab è `short e` il comando `format` senza nulla riporta il formato al default.

Gestione dell'output su video

Abbiamo tre modi per ottenere come output video il contenuto di una variabile

- nome variabile tasto invio
- funzione disp (per una sola variabile)
- funzione fprintf (per una o più variabili)

I primi due metodi sono sostanzialmente uguali: il secondo non riporta il nome delle variabile e segno uguale, mentre il primo si

```
>> a='pippo pluto e paperino';
>> a

a =

    'pippo pluto e paperino'

>> disp(a)
pippo pluto e paperino
>>
```

il comando fprintf

In alternativa a disp si può usare fprintf.

```
>> fprintf('Frankestein Junior')
Frankestein Junior>>
```

Alcune opzioni utili del comando fprintf:

\n permette di mandare a capo

```
>> fprintf('Frankestein Junior \n')
Frankestein Junior
>>
```

\t permette di indentare il testo che segue, ovvero spostare la stringa verso destra.

```
>> fprintf('\t Frankestein Junior \n')
        Frankestein Junior
>>
```

Specifica del formato di visualizzazione

Nel caso in cui la stringa contenga variabili numeriche, queste vengono inserite nella stringa indicandone il formato e si elencano poi alla fine della stessa. Il formato può essere decimale apunto fisso %f o esponenziale %e. Si possono inoltre specificare quante cifre dopo la virgola si vogliono rappresentare e quante cifre/spazi bianchi predisporre (utile per creare tabelle):

```
>> s=10.123456789;
>> fprintf('la variabile s vale %12.5e \n', s)
la variabile s vale  1.01235e+01
>> fprintf('la variabile s vale %22.5e \n', s)
la variabile s vale              1.01235e+01
>>
```

%12.5e indica che voglio rappresentare s in formato esponenziale, con 5 cifre dopo la virgola e 12 caratteri a disposizione.

Il comando `input`

In molti casi può essere utile interagire con l'utente per richiedere di inserire da tastiera determinate quantità che devono essere assegnate ad una variabile.

Il comando `input` scrive un testo a video, chiedendo all'utente di inviare un valore e premere il tasto "`return`". Vediamone un esempio.

```
>> a=input('Scrivi un numero intero positivo: ');
Scrivi un numero intero positivo: 5
>> b=input('Scrivi un altro numero intero positivo: ');
Scrivi un altro numero intero positivo: 6
>> fprintf('Il prodotto dei numeri inseriti e'':  %8.0f \n',a*b);
Il prodotto dei numeri inseriti e':       30
>>
```

Script

Se dobbiamo collezionare più comandi da eseguire la command window è troppo limitante, useremo gli script!

- si crea con l'editor di matlab
- si salva con qualsiasi nome (no spazi no caratteri speciali) ed estensione .m
- si deve sempre salvare prima di eseguire
- si esegue tramite tasto run dell'editor (attenzione al salvataggio) o scrivendo il nome dello script nella command window e premendo invio
- dopo ogni istruzione meglio andare a capo (anche se si può non fare)
- la visualizzazione dell'output dei comandi deve essere evitato (quando non necessario): usare ";" quando serve.

Buone idee:

- dare alle variabili nomi lunghi ma "esplicativi" del significato
- commentare i programmi con "%"