



# Laboratorio di Calcolo Numerico LEZ 1

## Introduzione al corso e a Matlab

Federico Piazzon

22 Marzo 2022

# Outline

- 1 Presentazione
- 2 Introduzione a Matlab
- 3 I numeri in Matlab
- 4 Formato di visualizzazione in Matlab
- 5 Precisione numerica in Matlab
- 6 Arrotondamento numerico in Matlab
- 7 Gestione dell'output su Command Window
- 8 Input da Command Window
- 9 Creare ed eseguire programmi Matlab

Mi presento. . .

Federico Piazzon

email [fpiazzon@math.unipd.it](mailto:fpiazzon@math.unipd.it)

Ufficio 330 Dip. Matematica (Torre Archimede) Canale

moodle [Calc.Num\\_prof. Piazzon](#)

- Primo contatto con ambiente MatLab
- Esperimenti che esemplificano i fenomeni visti a lezione
- Basi per un uso autonomo ed avanzato nei corsi dei prossimi anni e nella vita post-accademica

- Docente:

- Preparazione del materiale: **slides** e **m-files** (programmi) (troverete sempre TUTTO nel canale moodle) atto a
- Illustrare in pratica alcuni fatti teorici visti a lezione tramite esempi ed esperimenti.
- Spiegazione dettagliata dei programmi che utilizzeremo o scriveremo.
- Creazione di "homeworks" settimanali a valutazione automatica con **Matlab Grader** (facoltativi ma fortemente consigliati). Iscrizione automatica per gli utenti moodle registrati.
- **Ricevimento telematico** per tutti i dubbi, problemi e difficoltà. Si veda annuncio su canale moodle.

- Didattica di supporto **Dott. Raffaele Stefano Cattolico**  
Email: [raffaelestefano.cattolico@unipd.it](mailto:raffaelestefano.cattolico@unipd.it)
  - aiuto durante le esercitazioni
  - tutorato online (correzione esercizi)
- Studenti:
  - Loggarsi → creare cartella di lavoro della lezione → scaricare files forniti per la lezione
  - Seguire in modo attivo := provare a fare i propri programmi
  - Fare domande quando credete che possano essere di interesse comune o quando risulta poco chiaro

- Dovete avere una copia di Matlab installata e funzionante A CASA: seguire le indicazione **COME OTTENERE MATLAB** nel canale moodle
- Testate i programmi scritti a lezione.
- Confrontatevi con i colleghi e lavorate in gruppo, se emergono difficoltà contattatemi (ricevimento) o utilizzate la chat.

## Migliori AMICI

- Help di Matlab
- Dott. Cattolico
- Slides del corso
- Google e MathWorks
- Compagni di corso

## Consigli...

- meglio programmare che prendere appunti
- fare i **propri** programmi
- copiare e salvare i programmi del docente
- fare gli esercizi per casa.

Prova pratica **in presenza** (salvo imprevisti dovuti alla pandemia)

- Scrittura di programmi simili\* a quelli visti a lezione e nelle esercitazioni per casa o implementazione di altri algoritmi (specificati nel compito)
- Uso di base dell'ambiente Matlab
- Osservazione critica dei risultati alla luce della teoria

\*: Simili non significa uguali, significa che sono nello stesso spirito e per i quali è richiesto aver affrontato quelli proposti a lezione.



# Il nostro compagno di viaggio

**Matlab** = **Matrix Laboratory** è un ambiente di programmazione (software commerciale)

- calcolatrice ad alta precisione
- in grado di operare (efficientemente e in modo estremamente facile per l'utente) su matrici e vettori
- programmabile (= possibilità di scrivere complicate ricette di 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 prevede di lavorare in una **cartella di lavoro** e "non sa" cosa c'è fuori
- Pannelli/schermate:
  - **Command Window**: specie di calcolatrice che esegue operazioni "istantanee" su dati caricati in memoria
  - **Working Space**: elenco delle variabili in memoria
  - **Current Folder**: cartella di lavoro attuale e suo contenuto
  - **Editor**: è un editor di testo con suggerimenti e shortcut
- Matlab legge molti tipi di files l'estensione più importante è **".m"**
- I files .m sono di due tipi:
  - **script**: serie di operazioni sulle variabili in memoria (o create nello script) che modificano le variabili in memoria
  - **function**: "ricette" di calcolo da poter eseguire all'occorrenza su dati variabili (input) ma che modificano solo i dati in uscita (output)

# 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) assieme alla sua assegnazione tramite l'operatore di assegnazione " $=$ ". Es:  $A=5.2$  (si noti che  $=$  non è commutativo!)
- Non tutti i contenitori sono adatti ad un certo tipo di dati. Matlab in genere cambia tipologia di contenitore in base al contenuto.
- tipi principali:
  - double, single: floating point in precisione singola o doppia (lo standard è double)
  - int32, int64, uint32, uint64: interi con segno e senza rappresentati in 32 o 64 bit
  - char: stringa di caratteri
- per visualizzare i dati contenuti in una variabile basta scrivere il suo nome nella command window e premere invio.

# Operatori base

- = assegnazione
- == uguale logico
- + 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"
- .\* moltiplicazione componente per componente quando compatibile
- ; "non mostra output di operazione"
- sum() somma degli elementi di un vettore
- max(), min() massimo e minimo di vettore o per colonne di una matrice

- Una matrice o vettore è definito in matlab tramite parentesi **quadre** `[ ]`
- Gli elementi in riga sono separati da `,` oppure da **spazi**
- Le righe sono separate da `;`

ES:  $v=[1\ 2\ 3]$  è equivalente a  $v=[1,2,3]$

ES:  $A=[1\ 2\ 3;4\ 5\ 6]$  è equivalente a  $A=[1, 2, 3;4, 5, 6]$

In matlab si tende a memorizzare tutto in vettori e matrici perchè le operazioni su questo tipo di variabili sono ottimizzate.

**NB:** Metà della prossima lezione sarà dedicata a vettori e matrici.

# Creazione e lettura vettori/matrici

- $A(i, j)$  elemento di  $A$  con riga  $i$  e colonna  $j$  (si inizia da 1)
- $A(i_1 : i_2, j)$  vettore delle componenti  $A(i_1, j), A(i_1 + 1, j) \dots, A(i_2, j)$
- $A(i, j_1 : \text{end})$  vettore delle componenti  $A(i, j_1), A(i, j_1 + 1) \dots$  fino all'ultima colonna

Vettori e matrici si creano con

- risultato di operazioni matriciali
- funzioni di creazione matrici
- concatenazioni di blocchi
- cicli **for** su righe e colonne (inefficiente! Da fare solo se si è costretti)

# Operazioni su vettori e matrici

- ' trasposizione (ma anche inizio e fine di una stringa)
- \*,/ moltiplicazione e divisione per scalare:  $c*v$
- +,-, somma e sottrazione per componenti
- .\* moltiplicazione di due vettori **per componenti**
- **size(A),size(A,2)** dimensioni di A, seconda dimensione di A.

Funzioni utili per la creazione di vettori e matrici

- zeros(m,n) e ones(m,n) creano matrici  $m \times n$  di zeri o di uni. zeros(m) è come zeros(m,m).
- eye(m) crea matrice identica di ordine m
- $A=\text{diag}(v)$  crea matrice diagonale
- $v=\text{diag}(A,k)$  estrae k-esima sotto/sopra diagonale

# I numeri in Matlab

Matlab utilizza di default una rappresentazione numerica in **doppia precisione**, cioè via 64 bit, seguendo lo standard IEEE floating-point. Questo permette di rappresentare numeri macchina in valore assoluto nell'intervallo  $[\text{realmin}, \text{realmax}]$ .

Per illustrare questo, una volta raggiunta la workspace di Matlab, vediamo quanto siano `realmin` e `realmax`, digitando

```
1 >> realmin
2 ans =
3      2.2250738585072e-308
4 >> realmax
5 ans =
6      1.79769313486232e+308
7 >>
```



# I numeri in Matlab

Una quantità rilevante é la **precisione di macchina** ossia il più grande errore relativo compiuto nell'approssimare un numero reale che sta in  $[realmin, realmax]$  con un numero macchina.

È la minima costante  $eps$  t.c., se  $x \in [realmin, realmax]$  e  $f1(x)$  è la sua rappresentazione tramite numeri macchina allora

$$\left| \frac{f1(x) - x}{x} \right| \leq eps$$

In Matlab risulta

```
1 >> eps
2 ans =
3      2.2204e-16
4 >>
```

Matlab dispone di altre "quantità" (in realtà sono funzioni) interessanti predefinite.

- `pi` che é il floating point rappresentante  $\pi$ .
- `+Inf` che rappresenta **+infinito**.
- `-Inf` che rappresenta **-infinito**.
- `NaN` che rappresenta **not a number**.

# I numeri in Matlab II

```
1 >> -5/0
2 ans =
3     -Inf
4 >> +5/0
5 ans =
6      Inf
7 >> 0/0
8 ans =
9      NaN
10 >> Inf/Inf
11 ans =
12      NaN
13 >> Inf^Inf
14 ans =
15      Inf
```

# 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 window

```
1 >> help format
```

per vedere tutti i formati disponibili.

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

```
1 >> format long e
2 >> 0.1
3 ans =
4      1.0000000000000000e-01
5 >> format short e
6 >> 0.1
7 ans =
8      1.0000e-01
9 >> format long
10 >> 0.1
11 ans =
12      0.10000000000000000
13 >> format short
14 >> 0.1
15 ans =
16      0.1000
```

# 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`.

```
1 >> ad = 1.234567890112233445566778899
2 ad =
3     1.234567890112233e+00
4 >> as = single(ad)
5 as =
6     single
7     1.2345679e+00
```

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

```
1 >> ad2 = double(as)
2 ad2 =
3     1.234567880630493e+00
```

# Arrotondamento numerico in Matlab

Esistono dei comandi Matlab che ci permettono di arrotondare un risultato secondo diversi criteri:

<code>fix</code>	arrotondamento verso 0
<code>round</code>	arrotondamento verso l'intero più vicino
<code>floor</code>	arrotondamento verso $-\infty$
<code>ceil</code>	arrotondamento verso $+\infty$

```
1 >> a=3.7;  
2 >> round(a)  
3 ans =  
4      4  
5 >> fix(a)  
6 ans =  
7      3
```

```
1 >> ceil(a)  
2 ans =  
3      4  
4 >> floor(a)  
5 ans =  
6      3
```

# Gestione dell'output su video

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

- nome variabile **tasto invio**
- funzione `disp`
- funzione `fprintf`

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

```
1  >> a='pippo pluto e paperino';
2  >> a
3
4  a =
5
6      'pippo pluto e paperino'
7
8  >> disp(a)
9  pippo pluto e paperino
10 >>
```



# Gestione dell'output su video

il comando `fprintf`

In alternativa a `disp` si può usare `fprintf`.

```
1 >> fprintf('Frankenstein Junior')
2 Frankenstein Junior>>
```

Alcune opzioni utili del comando `fprintf`:

`\n` permette di mandare a capo

```
1 >> fprintf('Frankenstein Junior \n')
2 Frankenstein Junior
3 >>
```

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

```
1 >> fprintf('\t Frankenstein Junior \n')
2         Frankenstein Junior
3 >>
```

# Specifica del formato di visualizzazione I

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 a punto fisso **%f** o esponenziale **%e**. Si possono inoltre specificare quante cifre dopo la virgola si vogliono rappresentare e quante spazi predisporre (utile per creare tabelle).

```
1 >> s=10.123456789;  
2 >> fprintf('la variabile s vale %12.5e \n', s)  
3 la variabile s vale 1.01235e+01  
4 >> fprintf('la variabile s vale %22.5e \n', s)  
5 la variabile s vale 1.01235e+01  
6 >>
```

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

# Specifica del formato di visualizzazione II

- Il punto che separa parte decimale e mantissa conta come carattere
- $\%a.bf$  con  $a < a1 := b + 1 + l$  dove  $l$  è la lunghezza della mantissa del numero ha lo stesso effetto di  $\%a1.bf$ .

```
1  >> x=1000*pi;  
2  >> fprintf('x=%13.8f\n',x)  
3  >> fprintf('x=%14.8f\n',x)  
4  >> fprintf('x=%15.8f\n',x)  
5  >> fprintf('x=%16.8f\n',x)  
6  >> fprintf('x=%1.8f\n',x)  
7  x=3141.59265359  
8  x= 3141.59265359  
9  x=  3141.59265359  
10 x=   3141.59265359  
11 x=3141.59265359
```

# Il comando input

In molti casi può essere utile interagire con l'utente per richiedere determinate quantità

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

```
1 >> a=input('Scrivi un numero intero positivo: ');
2 Scrivi un numero intero positivo: 5
3 >> b=input('Scrivi un altro numero intero positivo: '
4         ');
5 Scrivi un altro numero intero positivo: 6
6 >> fprintf('Il prodotto dei numeri inseriti e':
7         '%8.0f \n',a*b);
8 Il prodotto dei numeri inseriti e':      30
9 >>
```

# Script

Per operazioni complesse la command window è troppo limitante, useremo gli script!

- si crea con l'editor di matlab (o con editor di testo)
- 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 questo prima salva!)  
o **scrivendo il nome dello script nella command window e premendo invio**
- dopo ogni istruzione meglio andare a capo
- l'output video deve essere il minimo necessario: usare ";" quando serve.

Buone idee:

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

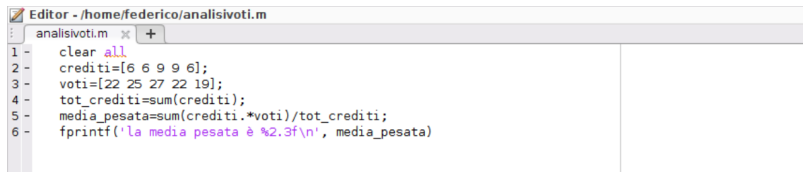
## Esercizio 1.1

Vogliamo avere un programma **analisivoti.m** che, dati i crediti degli esami sostenuti e il risultato degli stessi (memorizzati in due vettori all'interno del programma), fornisca come output video:

- 1 la mediana dei voti
- 2 la media dei voti
- 3 la media pesata dei voti
- 4 il voto massimo
- 5 il voto minimo

# SOLUZIONE (traccia):

Creiamo un nuovo script con il tasto "new" dell'editor (in alternativa usiamo un qualsiasi editor di testo) come il seguente e lo salviamo con il nome analisivoti.m nella cartella di lavoro



```
Editor - /home/federico/analysivoti.m
analysivoti.m
1 - clear all;
2 - crediti=[6 6 9 9 6];
3 - voti=[22 25 27 22 19];
4 - tot_credits=sum(crediti);
5 - media_pesata=sum(crediti.*voti)/tot_credits;
6 - fprintf('la media pesata è %2.3f\n', media_pesata)
```

scriviamo analisivoti nella command window e premiamo invio. Otteniamo

```
1 >> analisivoti
2 la media pesata é 23.250
3 >>
```

## Esercizio 1.2

Si crei (partendo da una copia di `analisivoti.m`) uno script `analisivoti_input.m` che richieda (si usi il comando `input`) all'utente del programma di inserire il vettore dei crediti e il vettore degli esiti.