

Lab. 1 - Introduzione a Matlab

Alcune informazioni su Matlab

Matlab è uno strumento per il calcolo scientifico utilizzabile a più livelli, dalla calcolatrice tascabile, alla simulazione ed analisi di sistemi complessi.

Il nome Matlab è una abbreviazione di Matrix-Laboratory: la struttura di base è la **matrice**: ogni quantità (variabile) viene trattata come una matrice di dimensioni $m \times n$.

Uno scalare reale è una matrice 1×1 .

- In Matlab non è necessario dichiarare esplicitamente all'inizio del lavoro una variabile in termini delle sue dimensioni e del tipo dei suoi coefficienti (interi, reali, complessi)
→ notevole semplificazione
- è già predefinito un ampio insieme di matrici elementari (matrice identità, matrice nulla...)
→ matrici più complesse possono essere costruite rapidamente partendo da queste matrici fondamentali
- sono predefiniti vari operatori algebrici fra matrici di uso comune, quali ad esempio somma, prodotto, elevamento a potenza, nonché il calcolo del determinante o del rango di una matrice;

- sono predefinite numerose funzioni primitive di uso generale, dette **built-in functions**. Esse permettono di risolvere problemi complessi, ad esempio il calcolo degli autovettori ed autovalori di una matrice, la risoluzione efficiente di sistemi lineari, oppure la ricerca degli zeri di una funzione.

Le raccolte di funzioni dedicate ad uno specifico argomento vengono dette **toolboxes**. La finanza, la statistica, l'analisi dei segnali e delle immagini sono alcuni dei campi a cui sono dedicati dei toolboxes di Matlab

Dove trovare dispense e informazioni su Matlab?

- sul sito ufficiale di Matlab www.mathworks.com sono disponibili numerosi manuali (in inglese) sia introduttivi che dedicati più approfonditamente ad aspetti specifici (programmazione, grafica, toolboxes...)
- sui siti di numerose università sono riportati tutorial ed esempi di problemi studiati con l'uso di Matlab

Matlab è un software a pagamento. Esiste un software gratuito, Octave, che ne riproduce buona parte delle funzioni fondamentali (con minime differenze di sintassi e una grafica un po' più povera). Octave può essere scaricato alla pagina web www.octave.org.

Per iniziare...

All'avvio di Matlab appare il prompt `>>`, ovvero la linea da cui digitare le istruzioni nello spazio di lavoro.

Il comando **demo** mostra degli esempi significativi di possibili applicazioni del software. Il comando **doc** introduce ad alcuni aspetti di base di Matlab e mostra quali pacchetti (toolboxes) siano installati nella versione in uso.

L'**help** di MATLAB permette di ottenere informazioni dettagliate su qualsiasi comando.

Ad esempio: **help sqrt** (oppure **doc sqrt**). Il solo comando **help** elenca gli argomenti per i quali è disponibile la guida, suddivisi in grandi aree tematiche (funzioni elementari, trattamento di matrici, grafica...)

Alcuni trucchi utili...

- è possibile richiamare "storicamente" i comandi precedentemente digitati nella sessione di lavoro usando i tasti \uparrow , \downarrow
- è possibile spostarsi lungo la linea di comando corrente e modificare la riga scritta utilizzando i tasti \rightarrow , \leftarrow
- è possibile completare un'istruzione già precedentemente digitata scrivendone le prime lettere e utilizzando poi il tasto \uparrow

Alcuni comandi fondamentali da conoscere...

- l'istruzione **diary mywork.dat** apre il file di testo mywork.dat nel quale viene trascritto (a partire da quel momento) il flusso delle istruzioni digitate (è una cronaca del lavoro svolto).
L'istruzione **diary off** interrompe la scrittura della cronaca e chiude il file mywork.dat
- l'istruzione **whos** elenca le variabili attualmente attive in memoria e dà alcune informazioni importanti sulle loro caratteristiche (tipo di oggetto, dimensioni in memoria..)
- l'istruzione **save area.mat** permette di salvare nel file binario area.mat il contenuto di tutte le variabili attive in memoria in quel momento.
- l'istruzione **save area.mat z x** salva le sole variabili **z** e **x**
- l'istruzione **load area.mat** ricarica le variabili salvate nel file area.mat e le rende attive in memoria (verificare con **whos**)
- il comando **quit** termina la sessione di lavoro e chiude Matlab.

Scalari in Matlab

In Matlab non è necessario definire e dichiarare le variabili. Tutte le variabili vengono trattate in **doppia precisione** (8 byte), senza distinzione fra interi, reali e reali a doppia precisione.

Iniziamo ad usare Matlab come una semplice calcolatrice: ad esempio scriviamo

```
>>z=3*2
```

assegnando così alla variabile **z** il valore 6.

Se scriviamo solamente

```
>>3*2
```

il valore 6 viene assegnato alla variabile **ans** (abbreviazione di answer). Tale variabile contiene sempre l'ultimo valore non esplicitamente assegnato dall'utente ad una variabile. Il **;** alla fine dell'istruzione sopprime la visualizzazione a schermo del risultato (ma non l'esecuzione effettiva dell'operazione!). Ad esempio, assegnamo alla variabile **a** il risultato di una certa operazione, senza visualizzarlo, e poi richiamiamo **a** (senza **;**) per vederne il valore

```
>>a=sqrt(100);
```

```
>>a
```

In Matlab vale la usuale precedenza fra operazioni, ad esempio la moltiplicazione (e divisione) ha precedenza sulla addizione (e sottrazione) e l'elevamento a potenza ha precedenza su addizione, sottrazione, moltiplicazione e divisione. Ad esempio:

```
>> 3+2*4
    11
>> 3*2^4
    48
```

Per alterare l'ordine delle operazioni ci si serve delle parentesi tonde. Anche quando non si vogliono alterare le precedenze, l'uso delle parentesi tonde è comunque sempre buona norma per chiarezza.

```
>> (3+2)*4
    30
>> (3*2)^4
    1296
```

Variabili predefinite: sono **pi** (pigreco), **i**, **j** (unità immaginarie), **eps** (epsilon macchina). Ogni variabile può essere tuttavia sovrascritta, ad esempio possiamo assegnare **pi=5** (attenzione!). Per cancellare il valore di una variabile (o se è predefinita portarla al suo valore iniziale) usiamo il comando **clear**. Ad esempio

```
>>pi
    3.1416
>>pi=5;
>> clear pi
>> pi
    3.1416
```

Il comando `clear all` cancella il valore di tutte le variabili (provare ad usare tale comando in combinazione con il comando `whos` che elenca le variabili presenti nello spazio di lavoro).

Se `a` e `b` sono due variabili scalari, abbiamo: la somma `a+b`, la sottrazione `a-b`, il prodotto `a*b`, la divisione `a/b`, la potenza `a^b`.

Vettori in Matlab

Per introdurre un **vettore riga** è sufficiente riportare i valori separati da spazi bianchi o virgole, ad esempio per introdurre $w \in \mathbb{R}^{1 \times 3}$:

```
>> w=[1 2 3]
```

oppure

```
>> w=[1, 2, 3]
```

Per introdurre un **vettore colonna** basta riportare fra parentesi quadre i valori della componenti del vettore stesso separati da un punto e virgola, ad esempio per introdurre $v \in \mathbb{R}^{3 \times 1}$:

```
>> 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:.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. La sintassi generale è `v=[valore_iniz:passo:valore_finale]`. Il passo può essere anche negativo, ad ex. `v=[10:-.5:1]`;

Il comando `linspace(valore_iniz, valore_fineale, N)` genera N valori equispaziati fra `valore_iniz` e `valore_finale` (estremi compresi). Ad esempio

```
>> v=linspace(0,1,5)
      0      0.2500      0.5000      0.7500      1.0000
```

Per **accedere alla componente di un vettore**, ad esempio alla terza, e assegnare alla variabile `z` tale valore, scriviamo `z=v(3)`.

Attenzione: la numerazione inizia da 1 e non da zero!

Nota: esistono in Matlab le parole chiave `start` e `end` per accedere rispettivamente al primo e all'ultimo elemento di un vettore. Ad ex., se `v` ha dieci elementi, `v(start)` equivale a `v(1)`, mentre `v(end)` equivale a `v(10)`.

Matlab produce un messaggio di errore quando si cerchi di accedere ad una componente non definita, ad esempio se `v` ha dieci elementi e vogliamo accedere a `v(11)`, oppure se vogliamo accedere a `v(0)` o a `v(-2)`.

Per controllare la dimensione di una variabile, usiamo il comando **size**, ad esempio **size(v)**. Questo comando è anche utile quando Matlab segnala un conflitto di dimensioni fra quantità che si vogliono manipolare.

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

Il comando **zeros(n,1)** produce un vettore colonna di dimensione n con elementi tutti nulli. Il comando **zeros(1,n)** produce un vettore riga di dimensione n con elementi tutti nulli. Il comando **ones(n,1)** (**ones(1,n)**) genera un vettore colonna (riga) con tutte le componenti pari a 1.

Operazioni su vettori componente per componente

- modulo di un vettore v , dato da $\|v\| = \sqrt{(v,v)}$: comando **norm(v)** (equivalente alla norma 2 del vettore: **norm(v,2)**)
- trasposizione di un vettore **w** di dimensione $n \times 1$: **w'** (dimensione $1 \times n$)
- prodotto componente per componente (attenzione: differente dal prodotto scalare!) $(v,w) = (v_1w_1, v_2w_2, \dots, v_nw_n)$. In Matlab tale operazione si esegue usando la sintassi "punto", ovvero **v.*w**. Se i due vettori non hanno la stessa dimensione, si genera un errore
- elevamento a potenza componente per componente: ex. vo-

gliamo calcolare il cubo di ciascuna componente. Usiamo la sintassi $\mathbf{v} . ^{\wedge} 3$, il cui risultato fornisce il vettore $(v_1^3, v_2^3, \dots, v_n^3)$

Operazioni fra vettori della stessa dimensione

Siano v, w vettori riga di \mathbb{R}^n , con componenti v_i e $w_i, i = 1, \dots, n$ rispettivamente. Si ha:

- somma algebrica $v + w = (v_1 + w_1, \dots, v_n + w_n)$. In Matlab tale operazione si esegue scrivendo $\mathbf{v+w}$
- prodotto scalare tra due vettori $(v, w) = (v_1w_1 + v_2w_2 + \dots + v_nw_n)$. In Matlab tale operazione tra vettori della stessa dimensione si esegue scrivendo $\mathbf{v*w}$

Matrici in Matlab (primi comandi)

Per assegnare le matrici

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

diamo i comandi, rispettivamente

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

```
>> B=zeros(2,3);
```

Posso calcolare

```
>> C=A+B;
```

```
>> D=A*B';
```

Inoltre

```
>> A= eye(5);  
>> B= rand(5);  
>> C= B-A;  
>> s=A(1,2)+C(3,3);
```

Istruzioni di controllo e istruzioni condizionali

Sintassi generale:

```
if (condizione1==true)
    istruzione1
    ...
elseif (condizione2==true)
    istruzione 2
    ...
else
    istruzione 3
    ...
end

for contatore = start:passo:end
    istruzione
    ...
    istruzione
end

while (condizione==true)
    istruzione
    ...
    aggiornamento condizione
end
```

Operatori logici

In Matlab gli operatori logici restituiscono il valore 1 se la condizione è vera, mentre restituiscono 0 se la condizione è falsa

- AND: &
- OR: ||
- a è uguale a b?: `a==b`
- a è diverso da b?: `a~=b`

Esempi di uso di istruzioni `if`, `for`, `while` con operatori logici:

```
>> n=5;
>> for i = 1:n
    if (i==1)|| (i==3)
        a(i) = 1/i;
    else
        a(i) = 1/((i-1)*(i-3));
    end
end
```

```
>> n=7;
>> for i = 1:n
    for j=1:n
        A(i,j)=1/(i+j-1);
    end
end
```

```

    end
end

>> n=10; i=1;
>> while(i<=n)
    if (i~=3)
        a(i) = 1/i;
    else
        a(i) = 1/(i-3);
    end
    i=i+1;
end

```

Funzioni simboliche

Esiste in Matlab una sintassi che permette di definire una funzione in modo simbolico. Tale potenzialità permette di manipolare agevolmente funzioni anche molto complesse e dipendenti da più parametri. Nella forma più semplice della sintassi, utilizziamo il comando **inline**, che definisce una funzione "in linea", ovvero direttamente nello spazio di lavoro, senza ricorrere ad un file esterno.

Per esempio, definiamo la funzione $f(x) = (\sin(x) + x)^2$:

```
>> f=inline('(sin(x)+x).^2','x')
```

dove abbiamo indicato esplicitamente che **f** è funzione di **x**. Attenzione alla sintassi con gli apici e i punti e attenzione all'operazione di elevamento a potenza componente per componente!

Ad una funzione così definita non sono associati dei valori numerici (verificare con **whos f**). Se ora vogliamo associare dei valori numerici, scriviamo

```
>> x=0:0.01:2*pi;  
>> y=f(x);
```

La sintassi **f(x)** permette di assegnare ad **f** dei valori numerici in corrispondenza degli elementi del vettore **x**. Tali valori numerici vengono conservati nel vettore **y** (verificare con **whos y**).

Possiamo per esempio disegnare il grafico di $y = f(x)$ con il semplice comando

```
>> plot(x,y)
```

Attenzione: perché il comando

```
>> plot(x,f)
```

non funziona?