

Università degli Studi di Ferrara

Corso di Laurea in Matematica - A.A. 2021 – 2022

Programmazione

Lezione 14 – Vettori e Matrici in MATLAB

Docente: Michele Ferrari - michele.ferrari@unife.it

Nelle lezioni precedenti

- MATLAB è un software creato da MathWorks e comprende un ambiente per il calcolo numerico e l'analisi statistica (scritto in C) e un linguaggio di programmazione. Il nome MATLAB è l'abbreviazione di MATrix LABoratory
- MATLAB nasce da delle librerie Fortran create per fornire accesso a LINPACK e EISPACK
- Caratteristica fondamentale in MATLAB è il fatto che l'elemento base è un array che non richiede dimensionamento: questo semplifica grandemente la manipolazione di tipi di dato vettoriali
- MATLAB è fornito agli studenti dell'Università di Ferrara tramite abbonamento annuale
- La command window permette una sessione interattiva, tramite essa è possibile creare variabili, utilizzare comandi e funzioni, eseguire operazioni

In questa lezione

- MATLAB: Vettori e Matrici
- Analisi dei tipi di dato in MATLAB
- Creazione di Vettori e Matrici
- Manipolazione di Vettori e Matrici

MATLAB: Vettori e Matrici

Negli esempi visti la scorsa lezione le variabili utilizzate apparentemente erano quantità scalari, ossia semplici valori numerici.

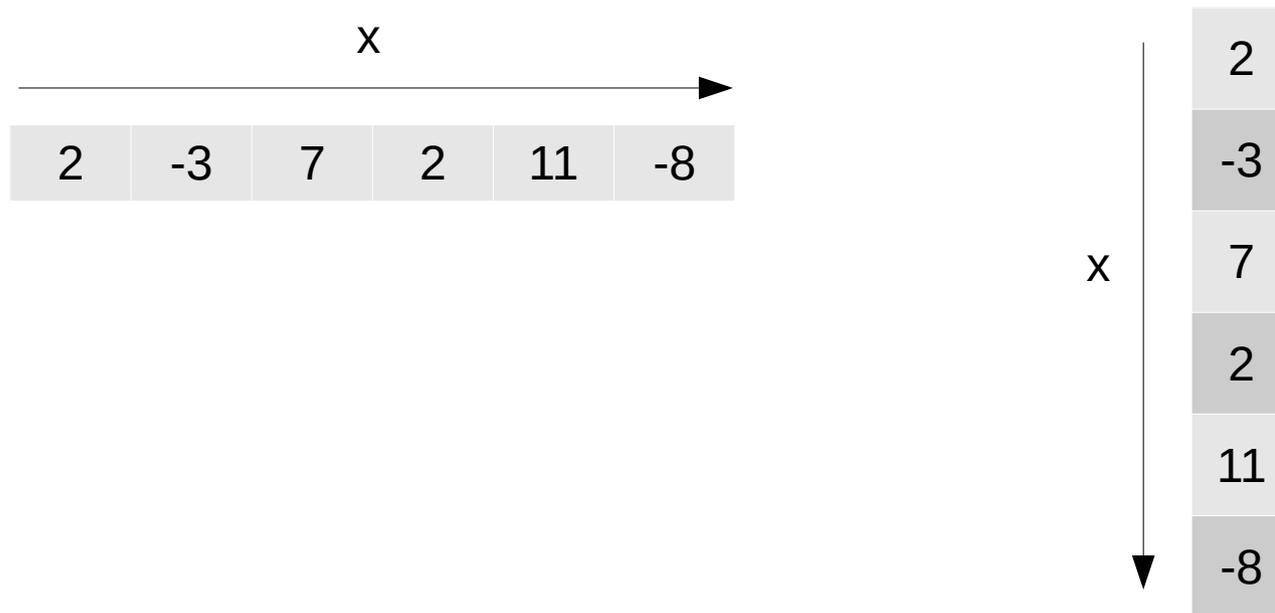
→ In realtà ogni variabile è, per MATLAB, una struttura dati di tipo vettoriale (o array).

Ricordiamo: Un array è un insieme di valori ordinati, secondo uno o più indici, cui ci si riferisce con un singolo nome di variabile.

Tipicamente un array ad un indice è detto **vettore**, ed un array a due indici è chiamato **matrice**.

Vettori riga e colonna

La sequenza di numeri interi $\{2, -3, 7, 2, 11, -8\}$ può essere rappresentata in forma di array ad un solo indice x nelle forme:



L'array di sinistra è detto **vettore riga**, quello di destra **vettore colonna**.

Entrambi hanno lunghezza 6. Tramite l'uso degli indici, **la cui numerazione va da 1 a 6**, da sinistra a destra nei vettori riga e dall'alto verso il basso nei vettori colonna, possiamo accedere ad un dato valore all'interno dell'array. Ad esempio a $x(2)$ corrisponderà il valore -3 , così come a $x(5)$ il valore 11.

Matrici

Potremmo anche memorizzare gli elementi come array bidimensionali a due indici nelle forme:

2	-3	7
2	11	-8

2	2
-3	11
7	-8

Ricordiamo che la matrice di sinistra è detta di tipo 2×3 , dove 2 indica il numero di righe e 3 il numero di colonne. Conseguentemente quella di destra è detta di tipo 3×2 .

Il numero di righe per il numero di colonne fornisce il numero di elementi della matrice, ossia 6.

Matrici

Analogamente a quanto visto in linguaggio C, per manipolare una matrice abbiamo bisogno di due indici, uno riferito alle righe e l'altro alle colonne.

Attenzione: in Matlab l'indicizzazione inizia da 1 (e non da 0 come in C)

In Matlab la sintassi per l'accesso al valore in riga 1, colonna 2 di una matrice A è:

```
>> A(1,2)
```

Ovvero si utilizzano le parentesi tonde per indicare gli indici e la virgola come separatore per indice di riga, indice di colonna

Matlab: gestire i dati

Di conseguenza i vettori x riga e colonna nell'esempio precedente sono rispettivamente matrici di tipo 1×6 e 6×1 : se uno scalare in matlab è una matrice 1×1 , un vettore in matlab è una matrice $1 \times M$ (vettore riga) o $N \times 1$ (vettore colonna).

Matlab: dichiarare un vettore riga

Per memorizzare il precedente vettore x nella forma riga in MATLAB possiamo utilizzare la seguente espressione:

```
>> x=[2 -3 7 2 11 -8]
```

```
x =
```

```
    2   -3    7    2   11   -8
```

Le parentesi quadre delimitano gli elementi del vettore, mentre gli spazi delimitano le singole componenti del vettore riga.

Matlab: dichiarare un vettore colonna

Per ottenere un vettore colonna è necessario utilizzare il delimitatore “;”

```
>> x=[2; -3; 7; 2; 11; -8]
```

```
x =
```

```
2
```

```
-3
```

```
7
```

```
2
```

```
11
```

```
-8
```

Matlab: accesso ai dati

Per visualizzare il valore di una componente del vettore basta scrivere

```
>> x(2)
```

```
ans =
```

```
-3
```

Se cambiamo il valore di una componente del vettore, dobbiamo ricordarci di usare il punto e virgola altrimenti MATLAB visualizzerà l'intero vettore:

```
>> x(2)= -7
```

```
x =
```

```
2
```

```
-7
```

```
7
```

```
2
```

```
11
```

```
-8
```

Matlab: Trasposizione

Per passare da vettori riga a vettori colonna si utilizza il simbolo di apostrofo, operazione che dal punto di vista dell'algebra lineare corrisponde alla trasposizione:

```
>> x=[2 -3 7 2 11 -8]'
```

```
x =
```

```
2
```

```
-3
```

```
7
```

```
2
```

```
11
```

```
-8
```

Matlab: funzione length

Qualora volessimo determinare la lunghezza di un vettore potremmo utilizzare la funzione length:

```
>> length(x)
```

```
ans =
```

```
6
```

Matlab: dichiarare una Matrice

Se ora vogliamo inserire la matrice 2×3 vista nell'esempio possiamo combinare i separatori appena visti per ottenere il risultato desiderato:

```
>> A=[2 -3 7; 2 11 -8]
```

```
A =
```

```
 2  -3  7
 2  11 -8
```

Notiamo che, anche in questo caso, **gli spazi separano gli elementi per colonna e il punto e virgola separa le righe.**

(Alternativa)La stessa matrice può anche essere inserita utilizzando il tasto Invio per separare le righe

```
>> A=[2 -3 7
```

```
 2 11 -8]
```

```
A =
```

```
 2  -3  7
 2  11 -8
```

Matlab: accesso ai dati 2

Potremo accedere agli elementi della matrice in maniera naturale utilizzando i corrispondenti indici:

```
>> A(1,2)
```

```
ans=
```

```
-3
```

(Ricordiamo che l'indicizzazione, in matlab, inizia da 1)

Se vogliamo quindi cambiare l'elemento A(1,2) basterà scrivere:

```
>> A(1,2)=-7
```

```
A =
```

```
2   -7   7
```

```
2  11  -8
```

prestando attenzione al fatto che senza punto e virgola finale viene visualizzata l'intera matrice.

Matlab: funzione size

La funzione MATLAB che ci consente di determinare le dimensioni di una matrice è size:

```
>> size(A)
```

```
ans =
```

```
     2     3
```

che restituisce **un vettore riga di due elementi interi**, il primo indica il numero di righe ed il secondo il numero di colonne.

Matlab: funzione size

Possiamo utilizzare la funzione size per verificare come Matlab memorizza e gestisce i tipi di dato:

```
>> a=3;
```

```
>> size(a)
```

```
ans =
```

```
    1    1
```

```
>> b=3+3i;
```

```
>> size(b)
```

```
ans =
```

```
    1    1
```

```
>> x=[2 -3 7 2 11 -8];
```

```
>> size(x)
```

```
ans =
```

```
    1    6
```

```
>> size(x')
```

```
ans =
```

```
    6    1
```

Visualizzazione variabili

- Per visualizzare in ogni momento il contenuto di una variabile sarà sufficiente scriverne il nome
- Per visualizzare il contenuto di più variabili sarà sufficiente scriverne i nomi separati da virgola
- Per visualizzare tutte le variabili presenti nel workspace si può utilizzare il comando **who**

```
>> who
```

```
Your variables are:
```

```
A    a    b    x
```

- Per una visualizzazione più esaustiva di tutte le variabili è possibile utilizzare il comando **whos**

```
>> whos
```

Name	Size	Bytes	Class
A	2x3	48	double array
a	1x1	8	double array
b	1x1	16	double array (complex)
x	1x1	48	double array

Grand total is 14 elements using 120 bytes

Notiamo che con whos oltre alle informazioni sulla dimensione e la lunghezza delle variabili MATLAB vengono fornite alcune indicazioni utili sullo spazio occupato in memoria dalle stesse variabili. Come si può notare una variabile non complessa come a occupa 8 bytes in memoria, mentre una variabile complessa come b occupa 16 bytes.

A e x, entrambi costituiti da 6 elementi non complessi, occupano $6 \times 8 = 48$ bytes in memoria.

Il comando clear

Per rimuovere le variabili dalla memoria lavoro di MATLAB si utilizza l'istruzione clear

```
>> clear
```

Provare `help clear` per avere una descrizione approfondita dell'uso del comando per rimuovere singole variabili e funzioni.

Inizializzare un array: la notazione due punti

Matlab fornisce diverse funzioni predefinite per la costruzione di vettori e matrici.

Un operatore fondamentale per costruire vettori equispaziati e per operare con indici è la notazione due punti.

Sintassi:

$$\text{Vettore} = \text{Inizio} : \text{Passo} : \text{Fine}$$

Dove

- Vettore: vettore riga
- Inizio: valore iniziale del vettore
- Fine: valore finale del vettore
- Passo: parametro opzionale che indica l'incremento relativo o la spaziatura tra gli elementi (default Passo=1)

Notazione due punti: esempi

```
>> x=1:10
```

```
x =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>> x=10:-1:1
```

```
x =
```

```
10 9 8 7 6 5 4 3 2 1
```

Notazione due punti: vettore colonna

Possiamo creare vettori colonna applicando alla notazione due punti la l'operatore di trasposta

```
>> y=(1:5)'
```

```
y =
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

Notiamo inoltre che se, con passo positivo, il valore iniziare è maggiore del valore finale verrà generato un vettore nullo:

```
>> x=10:1
```

```
x =
```

```
[]
```

Analogo risultato avremo con passo negativo e valore finale maggiore dell'iniziale. Il vettore nullo ha lunghezza zero, dimensione 0×0 ed è denotato da due parentesi quadre senza nulla all'interno.

E' possibile utilizzare la notazione due punti anche con valori non interi:

```
>> x=0:0.1:0.5
```

```
x =
```

```
0 0.1000 0.2000 0.3000 0.4000 0.5000
```

La funzione linspace

In alternativa se volessimo creare un vettore con un numero prefissato di punti equispaziati all'interno di un dato intervallo potremmo utilizzare la funzione linspace

Sintassi

```
linspace(Inizio, Fine, Numero di Punti)
```

Dove

- Numero di Punti: parametro opzionale che indica il numero prefissato di punti equispaziati desiderati, default=100

Linspace: esempio

Ad esempio

```
>> a = 0; b=1; n=5;
```

```
>> x = linspace(a,b,n)
```

```
x =
```

```
    0    0.2500    0.5000    0.7500    1.0000
```

restituisce un vettore riga x di lunghezza n con la proprietà che l'elemento di indice i vale

$$x(i) = a + (i - 1) * (b - a)/(n - 1)$$

In particolare le due istruzioni $x=linspace(a,b)$ ed $x=linspace(a,b,100)$ sono equivalenti.

Anche in questo caso vettori colonna possono essere costruiti tramite l'operatore di trasposizione:

```
>> a = 0; b=1; n=5;
```

```
>> x = linspace(a,b,n)';
```

```
>> x =
```

```
    0
```

```
 0.2500
```

```
 0.5000
```

```
 0.7500
```

```
 1.0000
```

Notazione due punti come definizione di intervallo

Un uso particolarmente efficace della notazione due punti si ha nella gestione di indici di vettori e matrici. In particolare tale notazione consente di identificare facilmente un'intera riga o colonna di una matrice:

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

```
>> A(:,1)
```

```
ans =
```

```
1
```

```
4
```

```
7
```

```
>> A(2,:)
```

```
ans =
```

```
4 5 6
```

Alternativamente è possibile specificare un intervallo di indici ed estrarre così parti di vettori (o matrici)

```
>> x=0:0.1:0.5;
```

```
>> x(2:4)
```

```
ans =
```

```
0.1000 0.2000 0.3000
```

Notazione due punti come definizione di intervallo

Selezionare parti di righe e colonne in matrici e sottomatrici:

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

```
>> A(2,2:3)
```

```
ans =
```

```
    5    6
```

```
>> A(1:2,2:3)
```

```
ans =
```

```
    2    3  
    5    6
```

La notazione due punti può essere usata anche per assegnare in modo rapido nuovi valori a righe e colonne di matrici

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

```
>> A(1,:)=2:2:6
```

```
A =
```

```
    2    4    6  
    4    5    6  
    7    8    9
```

Notazione due punti come definizione di intervallo

Possiamo adattare le tecniche appena viste per cancellare elementi da matrici o vettori cambiandone contemporaneamente la dimensione:

```
>> x = 1:10;
```

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

```
x =
```

```
 4  5  6  7  8  9 10
```

Possiamo anche rimuovere intere righe o colonne di una matrice:

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

```
>> A(:,1)=[]
```

```
A =
```

```
 2  3
```

```
 5  6
```

```
 8  9
```

Funzioni di inizializzazione

In tabella alcune funzioni MATLAB che consentono di costruire particolari matrici e vettori

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

Operazioni in MATLAB

Molte funzioni predefinite in MATLAB accettano come argomenti array a più indici. 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 a livello scalare.

Esempio (Tabella di valori di seno e coseno)

Per costruire una semplice tabella di valori delle funzioni seno e coseno nell'intervallo $[0, \pi]$ possiamo procedere nel seguente modo

```
>> n=5;
```

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

```
>> c=cos(x);
```

```
>> s=sin(x);
```

Operazioni in MATLAB

```
>> [x' c' s']
```

```
ans =
```

```
    0    1.0000    0
0.7854  0.7071  0.7071
1.5708  0.0000  1.0000
2.3562 -0.7071  0.7071
3.1416 -1.0000  0.0000
```

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(1)) cos(x(2)) cos(x(3)) cos(x(4)) cos(x(5))]`

Analogo discorso per l'istruzione vettoriale `s=sin(x)`. Infine l'istruzione `[x' c' s']` crea una matrice le cui colonne sono i vettori trasposti `x'`, `c'` e `s'`.

Operazioni in MATLAB

```
>> x = 1:5;
>> y = [50 10 30 40 20];
>> 2*x
ans =
     2     4     6     8    10

>> x+y
ans =
    51    12    33    44    25

>> y-x
ans =
    49     8    27    36    15

>> x.*y
ans =
    50    20    90   160   100

>> y./x
ans =
    50     5    10    10     4

>> y.^x
ans =
    50    100   27000  2560000  3200000
```

Operazioni in MATLAB

Le operazioni appena viste agiscono contemporaneamente su tutti gli elementi degli array considerati:

Le prime tre operazioni seguono le regole dell'algebra lineare numerica e sono la moltiplicazione di un vettore per uno scalare, la somma e la differenza tra vettori:

- Il comando $2*x$ moltiplica ogni componente di x per la quantità scalare 2.
(il vettore risultato ha esattamente la stessa lunghezza del vettore x)
- Il comando $x+y$ somma le rispettive componenti dei vettori x e y e può essere utilizzato solo su vettori che hanno la stessa dimensione.
- Il comando $y-x$ sottrae dal vettore y le corrispondenti componenti di x .
(entrambi i comandi restituiscono vettori aventi la stessa dimensione dei vettori argomento)

Operazioni in MATLAB

Le ultime 3 operazioni viste nell'esempio sono tipiche dell'ambiente Matlab e non hanno riscontro dal punto di vista dell'algebra lineare: in questi casi le matrici ed i vettori vanno intesi più come strutture dati che come entità matematiche.

Le tre operazioni in questione sono moltiplicazione puntuale, divisione puntuale ed elevamento a potenza puntuale:

- 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
- La divisione puntuale $y./x$ restituendo un vettore le cui componenti sono il risultato della divisione delle corrispondenti componenti di y per quelle di x
- Il comando $y.^x$ eleva ogni componente di y alla corrispondente componente di x

Operazioni in MATLAB

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.

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:

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

```
>> x+1
```

```
ans =
```

```
    2    3    4    5    6
```

```
>> 1-x
```

```
ans =
```

```
    0   -1   -2   -3   -4
```

```
>> 2./x
```

```
ans =
```

```
    2.0000    1.0000    0.6667    0.5000    0.4000
```

```
>> x.^2
```

```
ans =
```

```
    1    4    9   16   25
```

Grazie per l'attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

Per la parte introduttiva di MATLAB:

L. Pareschi, G. Dimarco “Introduzione a MATLAB”, corso di Laboratorio di Calcolo Numerico 2006