Matrix Laboratory

- Una delle prerogative di Matlab consiste nella possibilità di eseguire le operazioni fondamentali direttamente su matrici e vettori senza la necessità di implementarle attraverso cicli "for" che ne scandiscano tutti gli elementi.

- Inoltre è dotato di moltissime routine specializzate per il calcolo scientifico e in particolare per l'algebra lineare.

- Nel seguito useremo il termine array per riferirci indifferentemente ad un vettore o ad una matrice

- L'oggetto su cui lavora Matlab è un array.

  - Scalare → array 1x1

```
>> a=1
a =
        1
```

Costruzione di Array

- Per **assegnare matrici** o **vettori** si ricorre ai caratteri speciali **[ ]**

- Gli elementi di una stessa riga devono essere separati da spazi vuoti (o da virgole); le diverse righe devono essere separate dal punto e virgola (o dall'andata a capo).

  - Vettore riga → array 1xn

```
>> a = [1 2 3 4]
a =
        1 2 3 4
```

```
>> y = [1,2,3,4];
y=
        1 2 3 4
```

  - Vettore colonna → array nx1

```
>> b = [1;2;]
b =
        1
        2
```

- I vari elementi possono essere valori costanti, nomi di variabile già assegnati o espressioni

```
>> a = 2;
>> b = 3;
>> x = [1 a 1 b]
x =
        1 2 1 3
```

- Per assegnare ad una variabile un vettore colonna si può far seguire all'assegnazione di un vettore riga il carattere apice, che rappresenta l'operatore di trasposizione:

```
>> x = [1 a 1 b]'
x =
        1
        2
        1
        3
```

- L'i-esimo elemento di un vettore è individuato da x(i) e l'ultimo elemento è individuato da x(end);

- Nell'assegnazione di un array il carattere spazio è sempre interpretato come il separatore tra due elementi. Si osservino ad esempio le seguenti istruzioni:

```
>> x = [1 2 3+4]
x =
        1 2 7

>> y = [1,2,3+4]
y =
        1 2 7

>> z = [1,2,3 +4]
z =
        1 2 3 4
```

- Lo stesso principio vale per assegnare delle matrici: caratteri su una stessa riga si separano con uno spazio o una virgola; le righe si terminano con un carattere ; o andando a capo.

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

- E' possibile assegnare matrici rettangolari con un numero arbitrario di righe e di colonne; l'unico vincolo è che ogni riga (e ogni colonna) deve avere la stessa lunghezza.

```
>> B = [ 1 2; 3]
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

- L'operatore di trasposizione può ovviamente essere usato anche per matrici.

```
>> A = [1 2; 3 4], B = A'
A =
        1 2
        3 4
B =
        1 3
        2 4
```

- Per indicare l'elemento ij di una matrice, si usa la sintassi nomematrice(i,j).

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

- Il comando **length(x)** fornisce le dimensioni di un array secondo la seguente sintassi:

```
>> length(x)
ans =
          2
```

Il comando **size(x)** fornisce le dimensioni di un array, che possiamo memorizzare in un vettore scrivendo:

```
>> [n,m] = size (b)
```

Assegnazione a blocchi

- L'assegnazione a blocchi è analoga a quella per elementi; ad esempio siano assegnati nel workspace i seguenti vettori:

```
>> x = [1 2];
>> y = [3 4];
```

- Ogni "elemento" può a sua volta essere un array, anziché un valore scalare, nel rispetto delle dimensioni complessive dell'array, si possono combinare a piacimento scalari, vettori e matrici.

```
>> z = [x y x]
z =
        1 2 3 4 1
```

- Per separare i blocchi su una stessa riga si usa sempre una virgola o uno spazio

```
>> B = [x; y]
B =
        1 2
        3 4
```

- Per separare blocchi su righe diverse si usa un punto e virgola o si va a capo.

```
>> C = [y' B; 5 6 7]
C =
        3 1 2
        4 3 4
        5 6 7
```

Notazione colon ":"

- Con la notazione **colon** vengono generati vettori di valori equispaziati compresi fra due valori estremi assegnati.

a:b

tale comando genera il vettore (a, a + 1, a + 2, ..., a + m), dove m è la parte intera di b – a. Se a > b, il comando genera un vettore vuoto.

```
>> x = 3:6
x =
        3 4 5 6

>> y = -4:2.2
y =
        -4 -3 -2 -1 0 1 2
```

- Più in generale, si può specificare la spaziatura da usare tra i valori:

a:d:b

genera il vettore (a, a+d, a+2d, ..., a+md), con m parte intera di (b–a)/d; d può essere arbitrario (< 0 e non intero);

```
>> x = 3:2:9
x =
        3 5 7 9

>> y = 3:2:10
y =
        3 5 7 9

>> z = 20:-5:0
z =
        20 15 10 5 0

>> w = 0:0.1:0.5
w =
        -0.0 0.1 0.2 0.3 0.4 0.5
```

- se a > b e d > 0 o se a < b e d < 0 viene generato un vettore vuoto

```
>>w = 5:0.1:0
w =
        Empty matrix: 1-by-0

>>v = [ ]
v =
        [ ]
```

- un array può ovviamente venire assegnato anche elemento per elemento, utilizzando uno o più cicli for come nella maggior parte dei linguaggi di programmazione.

- Quando un vettore sia costituito da elementi con differenza costante si può assegnare nel seguente modo:

nome_vettore= [valore_iniziale : incremento: valore_finale]

- Anche il comando **linspace** crea un vettore riga di **n** elementi linearmente intervallati, estremi compresi, la sintassi è:

linspace (valore_iniziale , valore_finale , n)

La differenza con colon è che a linspace non si specifica il passo ma il numero totale di punti che si desidera (di default **n=100**)

- Il comando **logspace** genera un array di elementi intervallati logaritmicamente, se n è il numero di punti tra 10^a e 10^b la sintassi è:

logspace (a, b, n)

Matrici Speciali

- eye(n) genera la matrice identità di ordine n

eye (n, {m}) eye (size(A))

```
> > B = eye(2)
B =
        1 0
        0 1
```

- zeros(m,n) genera una matrice di m righe e n colonne i cui elementi sono tutti uguali a 0

zeros (n, {m}) zeros (size(A))

```
> > B = zeros(2,3)
B =
        0 0 0
        0 0 0
```

- ones(m,n) come zeros, ma tutti gli elementi della matrice sono uguali a 1

ones(n, {m}) ones (size(A))

```
>> ones(2,2)
ans =
        1 1
        1 1
```

Ai comandi zeros, ones, è possibile passare anche un solo argomento; in questo caso si genera una matrice quadrata che ha per ordine l'argomento in ingresso;

```
>> ones(2)
ans =
        1 1
        1 1
```

per ottenere un vettore di n componenti occorre ad esempio usare il comando zeros(1,n) o zeros(n,1)

```
>> zeros(1,3)
ans =
        0 0 0
```

- rand(n,m) genera un vettore o una matrice di numeri casuali

rand(n, {m}) rand(size(A))

```
>> ones(2,2)
ans =
        1 1
        1 1
```

Sottomatrici

- Una volta che un array è assegnato nel workspace, è possibile fare riferimento ad una qualunque sua sottomatrice in modo estremamente agevole.

- Per estrarre da un array A di m righe e n colonne soltanto certi elementi che formano una sottomatrice, è sufficiente usare la sintassi seguente:

  A(ir,ic)

  dove ir ed ic sono due vettori contenenti gli indici che corrispondono, rispettivamente, alle righe e alle colonne da selezionare per estrarre la sottomatrice.

- la notazione colon (:) è utilizzata anche per la creazione di sottomatrici:

```
B = A(1:4, 3)
B = A(1:4, :)
```

```
B = A(:, 3)
B = A(:, [2 4 ])
```

Per estrarre la sottomatrice di testa di ordine 2 della matrice A così definita:

```
>> A=[1 2 3 ; 4 5 6; 7 8 9];
>> A([1 2],[1 2]), A(1:2,1:2)
ans =
          1 2
          4 5
```

Per estrarre una sottomatrice costituita dalle prime due colonne di A ma contenente tutte le righe

```
>> A(1:3, 1:2), A(:,1:2)
ans =
          1 2
          4 5
          7 8
```

Per rappresentare solo la seconda colonna di A

```
>> A(:,2)
ans =
          2
          5
          8
```

Per costruire una matrice B di 4 righe e 3 colonne avente le prime tre righe tutte uguali alla seconda riga di A e la quarta riga uguale alla prima riga di A

```
>> B = A([2 2 2 1],:)
B =
          4 5 6
          4 5 6
          4 5 6
          1 2 3
```

- Per rimuovere alcune componenti di un array si può procedere assegnando ad essi l'array vuoto [ ];

```
>> B(1:2,:) = [ ]
B =
4 5 6
1 2 3
```

- Un tipo di notazione analogo vale per i vettori. Sia v un vettore (riga o colonna, indifferentemente) di n elementi.

```
>>w = v(1:5)
```

crea un vettore w di cinque elementi uguali ai primi 5 elementi di v

```
>>z = v(n-4:n)
```

crea un vettore z i cui elementi saranno gli ultimi 5 elementi di v.

Matrici Speciali

- diag: se applicata ad una matrice estrae la k-ma diagonale della matrice

diag (A, {k})

```
>> a=[1:1:3;3:1:5;3:-1:1]
a =
        1 2 3
        3 4 5
        3 2 1
>> b=diag(a)
b=
        1
        4
        1
```

- diag: se applicata ad un vettore genera una matrice quadrata con k-ma diagonale uguale al vettore

diag (v, {k})

```
>> v=[1 2 3];
>> A=diag(v)
A=
        1 0 0
        0 2 0
        0 0 3

>> B=diag(v,1)
B=
        0 1 0 0
        0 0 2 0
        0 0 0 3
        0 0 0 0
```

- tril: estrae da una matrice la parte triangolare inferiore (lower) a partire dalla k-ma diagonale

tril (A, {k})

```
>> a=[1:1:3;3:-1:1;0:1:2]
a =
    1 2 3
    3 2 1
    0 1 2

>> b=tril(a)
b=
    1 0 0
    3 2 0
    0 1 2

>> c=tril(a,1)
c=
    1 2 0
    3 2 1
    0 1 2
```

- triu: estrae da una matrice la parte triangolare superiore (upper) a partire dalla k-ma diagonale

triu (A, {k})

```
>> a=[1:1:3;3:-1:1;0:1:2]
a =
    1 2 3
    3 2 1
    0 1 2

>> b=triu(a)
b=
    1 2 3
    0 2 1
    0 0 2

>> c=triu (a,-1)
c=
    1 2 3
    3 2 1
    0 1 2

>> c=triu (a,1)
c=
    0 2 3
    0 0 1
    0 0 0
```