Vettori e Matrici

- Vettori e matrici:
  - Creazione
  - Matrici particolari
  - Vettori regolarmente intervallati
  - Coordinate
  - Operatori utili
- Vettori di caratteri
- Polinomi

Vettori

- Unità fondamentale in Matlab
  - Tutte le variabili sono considerate matrici
- I valori sono organizzati in righe e colonne

$$dati = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix} \qquad name = \begin{bmatrix} Marty \\ James \\ Bob \end{bmatrix}$$

  - Numeri o stringhe
  - Ogni elemento ha posizione e valore

- Dimensione (Size) ≡ numero di righe e di colonne della matrice

  scritto come R per C o R x C

  dove R = numero di righe

  C = numero di colonne

  e.g.

$$dati = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$ yield è 3 per 4

$$test = \begin{bmatrix} 4 & 5 & 3 & 5 & 0 \end{bmatrix}$$ test è 1 per 5

- Le dimensioni particolari
  - scalare: vettore 1 x 1, es. 4 or [4]
  - Vettore riga: vettore 1 x C, es.

    [9 7 5 4 2] vettore riga 1 x 5
  - Vettore colonna: vettore R x 1, es.

$$\begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$$ vettore colonna 3 x 1

matrice: R x C con R > 1, C > 1

- se R = C → matrice *quadrata*

$$\begin{bmatrix} 4 & 5 & 3 \\ 10 & 4 & 66 \\ 18 & -3 & 2 \end{bmatrix}$$

- se R = C = 0 → matrice *vuota* [ ] (un paio di quadre vuote)

Creazione dei vettori

- Specifica diretta:
  - Nome seguito dall'operatore di assegnamento (=), come le variabili
  - Elenco dei valori dentro parentesi quadre ([ ])
  - Si inserisce una riga per volta
    - Da sinistra a destra, dall'alto in basso
    - Spazio o virgola tra i valori
    - Righe separate da punto e virgola o da invio

- Per esempio per:

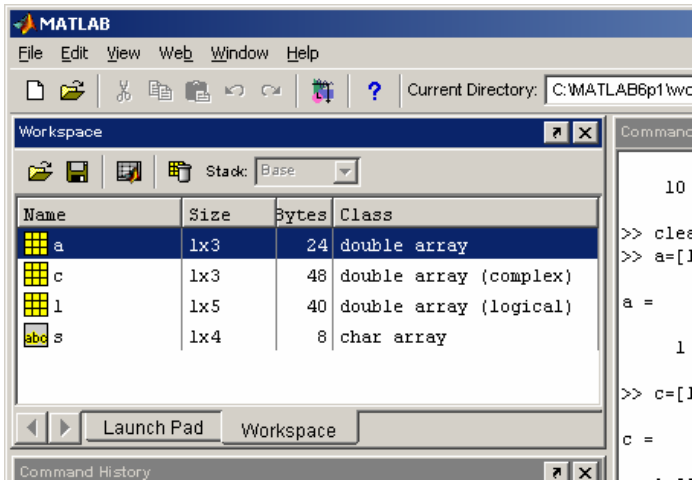
$$b = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$

usiamo

b = [4,5,3,9; 10,4,66,20; 18,-3,2,0] *enter*

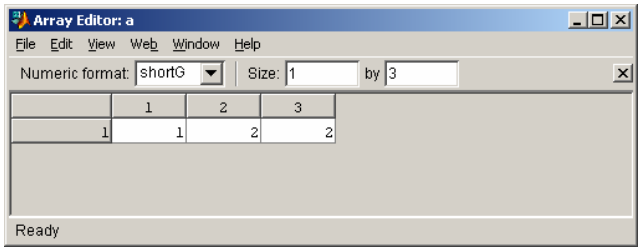
o

b = [4, 5, 3, 9 *enter*
10, 4, 66, 20 *enter*
18, -3, 2, 0] *enter*



Dimensioni e tipo dei vettori sono visibili nella finestra *workspace*

- Facendo doppio click su di una variabile dalla finestra *workspace* si apre l'*array editor*



- Possiamo modificare dimensioni e valori della matrice o del vettore

- Da command window, oltre ai numeri, possiamo usare costanti o operazioni in inserimento:

```
Command Window
>> b = [ 1000 exp(1) log10(100) 4+pi 2^-12 ]

b =

  1.0e+003 *

    1.0000    0.0027    0.0020    0.0071    0.0000
```

- Notare il formato comune per tutti i valori (exp(1) = *e* = 2.71828, $\log_{10}(100) = 2$, $2^{-12} = 0.00024414$) – MATLAB scala tutti gli esponenti verso il più alto!!!

- Certe volte questo è fuorviante:

```
Command Window
>> a = [ 3000*pi 3 .03 ]

a =

  1.0e+003 *

    9.4248    0.0030    0.0000

>> b = [ 3000*pi 3 0 ]

b =

  1.0e+003 *

    9.4248    0.0030         0
```

Non è un vero zero

Veramente zero

- Array speciali predefiniti:

Versioni quadrate

  - Matrice piena di 0
    zeros(R,C) zeros(N)
  - Matrice piena di 1
    ones(R,C) ones(N)
  - Matrice identità (tutti 0, 1 sulla diagonale)
    eye(R,C) eye(N)
  - Matrice di numeri casuali (nell'intervallo [0 1])
    rand(R,C) rand(N)

```
Command Window
>> eye(3,4)

ans =

     1     0     0     0
     0     1     0     0
     0     0     1     0

>> rand(2,5)

ans =

    0.9501    0.6068    0.8913    0.4565    0.8214
    0.2311    0.4860    0.7621    0.0185    0.4447
```

Casuale in [0, 1]

- Concatenare i vettori

se a = [1 2 3] b = [4 5 6]

- Attaccare a destra – usare la virgola

[a, b] → [1 2 3 4 5 6]

virgola

- Attaccare sotto – usare punto e virgola

[a; b] → $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Punto e virgola

- Le dimensioni devono combaciare!!!

se a = [1 2 3]
allora b = $\begin{bmatrix} 4 & 5 \\ 10 & 4 \end{bmatrix}$

[a, b] = ?? [a; b] = ??

- Per la concatenazione devono coincidere:
  - # di righe per affiacare (virgola)
  - # di colonne per incolonnare (punto e virgola)
- Possiamo usarli anche per fare “crescere” una matrice, es.

a = [1 2 3] a = [a , 4]

- Vettori regolarmente intervallati (: :)

primo : incremento : massimo

produce un vettore di elementi regolarmente intervallati

- Esempi:

0 : 2 : 10 → [0 2 4 6 8 10]
1 : 5 → [1 2 3 4 5]
7 : -2 : -3 → [7 5 3 1 -1 -3]
1 : 2 : 8 → [1 3 5 7]

- L’incremento di default è 1

Nota – non arriva ad 8!!

- Linspace – simile all'operatore due punti, ma arriva assolutamente all'estremo superiore

  linspace (start, last, numero di valori)

  - esempi:

    linspace(0,10,6) ⟶ [0 2 4 6 8 10]

    linspace(0,1,4) ⟶ [0 0.333 0.667 1]

  - Il default per il numero di valori è 100

    linspace(0,10) ⟶ [0 0.101 0.202 … 10]

    100 punti, 99 intervalli

- Logspace – crea un vettore di elementi intervallati *logaritmicamente.*

  logspace (a, b, numero di valori)

  I valori vanno da 10^a a 10^b

  - Ad esempio:

    logspace(-1,1,4)

    [0.1000, 0.4642, 2.1544, 10.000]

  - Il default per il numero di valori è 50

Coordinate nelle matrici

- Indichiamo un particolare elemento di una matrice specificandone la riga e la colonna:
- Usare le parentesi e la virgola
  - es.

dati(2,4)

$$dati = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$

- Per leggere un valore da una matrice
  - La selezione va a destra di =

```
test =

    0.4565    0.8214    0.6154
    0.0185    0.4447    0.7919

>> x = test(1,3)

x =

    0.6154
```

- Per cambiare un valore nella matrice
  - La selezione va a sinistra di =

```
test =

    0.4565    0.8214    0.6154
    0.0185    0.4447    0.7919

>> test(1,2) = 5

test =

    0.4565    5.0000    0.6154
    0.0185    0.4447    0.7919
```


- E se vogliamo selezionare più di un elemento?
  - Possiamo specificare una sottomatrice rettangolare
  - Usare le parentesi dopo il nome della matrice
  - Righe desiderate, virgola, colonne desiderate
  - come vettori separati, generalmente in quadre

es.

$$dati = \begin{bmatrix} 4 & 5 & 3 & 9 \\ 10 & 4 & 66 & 20 \\ 18 & -3 & 2 & 0 \end{bmatrix}$$

dati([1 2],[3 4])

- Esempio

```
sample =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

sample(3,2)
sample (1,2:4)
sample (3,[4,3,2,1])
sample ([1,3],[2,4])

- Per leggere una sottomatrice (a dx di =)

```
test =

    0.9218    0.1763    0.9355
    0.7382    0.4057    0.9169

>> x = test(1,[2 3])

x =

    0.1763    0.9355
```

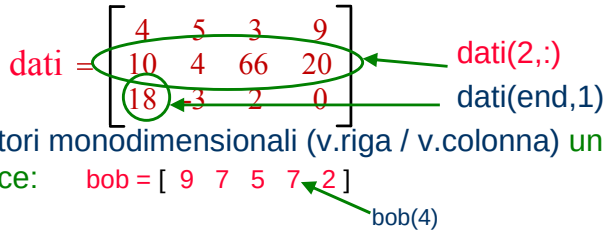
Per una sola riga non occorrono le quadre!

- Per cambiare una sottomatrice (a sx di =)

```
test =

    0.9218    0.1763    0.9355
    0.7382    0.4057    0.9169

>> test([1 2],1) = zeros(2,1)

test =

         0    0.1763    0.9355
         0    0.4057    0.9169
```

N.B. Deve avere la stessa dimensione !!!

- Ulteriori informazioni:
  - *end* identifica l'ultimo elemento della riga/colonna
  - Due punti (:) identificano tutta la riga/colonna



  - Per i vettori monodimensionali (v.riga / v.colonna) un solo indice: bob = [9 7 5 7 2]

bob(4)

  - N.B. Gli indici (o i vettori di indici) possono essere anche variabili, es:

index=[2,4]; bob(index)

- Errori comuni:

```
test =

    0.4103    0.0579    0.8132
    0.8936    0.3529    0.0099

>> test(2,4)
??? Index exceeds matrix dimensions.

>> test(0,-1.5)
??? Subscript indices must either be real
              positive integers or logicals.
```



```
test =

    0.4103    0.0579    0.8132
    0.8936    0.3529    0.0099

>> test(2,:) = [ 1 2 ]
??? In an assignment A(matrix,:) = B, the number
must be the same.

>> test(:,2) = [ 1 2 ]
??? In an assignment A(:,matrix) = B, the number
of columns in B must be the same.
```

Numero di elementi sbagliato nell'assegnamento

Dimensioni sbagliate (2x1 o 1x2 ?)

- Altri metodi utili:

```
test =

    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289

>> test(5) = 100

test =

    0.8462    0.6721    0.6813    0.5028
    0.5252  100.0000    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289
```

Se ci posizioniamo in una matrice usando un solo indice, si va per colonne!!!


```
test =

    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289

>> test(4,2) = 100

test =

    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289
         0  100.0000         0         0
```

Un assegnamento con un indice fuori misura ingrandisce la matrice!


```
test =

    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289

>> test([2 3],[2 3]) = 7

test =

    0.8462    0.6721    0.6813    0.5028
    0.5252    7.0000    7.0000    0.7095
    0.2026    7.0000    7.0000    0.4289
```

Usando un solo numero possiamo assegnare molti elementi


```
test =

    0.8462    0.6721    0.6813    0.5028
    0.5252    0.8381    0.3795    0.7095
    0.2026    0.0196    0.8318    0.4289

>> test(:,2) = []

test =

    0.8462    0.6813    0.5028
    0.5252    0.3795    0.7095
    0.2026    0.8318    0.4289
```

Sostituire una riga o una colonna (INTERE!!) con una matrice vuota, vuol dire eliminarle

- Riassumendo:
  - Simboli da usare:
    - Parentesi quadre per raggruppare elementi in una matrice
    - virgola (o spazio) e punto e virgola (o enter) per separare gli elemnti in righe/colonne
    - matrice(riga,colonna) per selezionare
  - Attenzione alle dimensioni (contano!)

Operatori utli per le matrici

- Trasposta (virgoletta singola ')
  - Scambia righe e colonne

```
test =

     1     2     3     4
     5     6     7     8
     9    10    11    12
```

```
>> test'

ans =

     1     5     9
     2     6    10
     3     7    11
     4     8    12
```


- Size – il numero di righe e di colonne
- Length – il maggiore di questi

$$test = \begin{bmatrix} 4 & 5 & 3 \\ 10 & 4 & 66 \end{bmatrix} \qquad bob = [\ 5\ 7\ 3\ 6\ ]$$

```
>> size(test)

ans =

     2     3

>> length(test)

ans =

     3
```

Risultato vettore

Risultato scalare

```
>> size(bob)

ans =

     1     4

>> length(bob)

ans =

     4
```


- Diag – operatore matrice ⟺ vettore per gli elementi diagonali

```
>> diag([ 1 2 3 ])

ans =

     1     0     0
     0     2     0
     0     0     3
```

```
test =

    0.6449    0.3420    0.5341    0.8385
    0.8180    0.2897    0.7271    0.5681
    0.6602    0.3412    0.3093    0.3704

>> diag(test)

ans =

    0.6449
    0.2897
    0.3093
```

- Reshape
  - Ridimensiona l'insieme di elementi

```
test =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>> reshape(test,2,6)

ans =

     1     9     6     3    11     8
     5     2    10     7     4    12
```

N.B. Gli elementi rimangono gli stessi

Riordinamento seguendo le colonne!

Creare matrici velocemente

- Per creare una matrice come:

```
test =

     1     2     3     4
     5     6     7     8
     9    10    11    12
```

- Bisogna:
  - Creare un elenco da 1 a 12
  - Fare il reshape in 4x3
  - Trasporre
- In pratica:
  - A=1:12
  - T=reshape(A,4,3)
  - test = T'
- O in un unico comando:
  - test = reshape(1:12 , 4, 3) '

Vettori di caratteri

- Le stringhe di caratteri alfanumerici sono vettori di caratteri.
- Un elemento per carattere!
- Usare la virgoletta singola (') prima e dopo

```
>> test = 'John'

test =

John

>> size(test)

ans =

     1     4
```


- Per vettori alfanumerici di più righe, il numero di caratteri deve essere lo stesso!

  name = ['Marty' ; 'James' ; 'Bob ']

  N.B. – ci sono 2 spazi

- Per un gruppo di parole di lunghezze diverse si usa il vettore di celle

  name = { 'Marty' ; 'James' ; 'Bob' }

- Per inserire un apostrofo bisogna metterne due!

  'Mary''s' → Mary's (vettore 1 per 6)

- Esistono ache dei vettori predefiniti:

  y = date → y = 05-Jan-2004

  (vettore 1 per 11)

Polinomi

- In MATLAB i vettori vengono utilizzati spesso per rappresentare i coefficienti di un polinomio.
- Ad esempio i coefficienti di:

  $x^3 - 7x^2 + 40x - 34 = 0$

  sono rappresentati nel vettore

  `a = [1,-7,40,-34]`
- In ordine decrescente!!!

Prodotto e divisione tra polinomi

La funzione `conv(a,b)` calcola il prodotto dei due polinomi descritti dai vettori dei coefficienti a e b. I due polinomi possono non avere lo stesso grado. Il risultato è il vettore dei coefficienti del prodotto polinomiale.

La funzione `[q,r] = deconv(num,den)` calcola il risultato della divisione tra un polinomio numeratore (il cui vettori dei coefficienti è `num`) ed un polinomio denominatore (il cui vettori dei coefficienti è `den`). Il polinomio quoziente è dato dal vettore dei coefficienti `q`, e il vettori dei coefficienti del resto è `r`.

Esempio

```
>>a = [9,-5,3,7];
>>b = [6,-1,2];
>>product = conv(a,b)

product =
   54   -39   41   29   -1   14

>>[quotient, remainder] = deconv(a,b)

quotient =
   1.5   -0.5833
remainder =
   0   0   -0.5833   8.1667
```


Radici di un polinomio

La funzione `roots(a)`calcola le radici di un polinomio definito dal vettore dei coefficienti a. Il risultato è un vettore colonna che contiene le radici del polinomio.

Per esempio:

```
>>r = roots([2, 14, 20])
r =
   -5
   -2
```

Viceversa: otteniamo i coefficienti

La funzione `poly(r)` calcola i coefficienti del polinomio le cui radici sono specificati nel vettore `r`. Il risultato è un vettore riga che contiene i coefficienti del polinomio.

Per esempio:

```
>>c = poly([-5, -2])
c =
    1    7    10
```


Valori del polinomio

La funzione `polyval(a,x)` calcola i valori del polinomio per i valori specificati nel vettore (o matrice) `x`, per la sua variabile indipendente. I coefficienti del polinomio sono contenuti nel vettore a. I risultati hanno le stesse dimensioni di `x`.

(Differential engine di Babbage)

Grafico del polinomio

Per disegnare il polinomio

$f(x) = 9x^3 - 5x^2 + 3x + 7$ per $-2 \le x \le 5$

Digitiamo:

```
>>a = [9,-5,3,7];
>>x = [-2:0.01:5];
>>f = polyval(a,x);
>>plot(x,f),xlabel('x'),ylabel('f(x)')
```


Grafici

- plot(x,y) dove x e y sono vettori di valori
- Per mettere le etichette:
  - title('Qua va il titolo')
  - xlabel('Etichetta asse X')
  - ylabel('Etichetta asse Y')
- Altri comandi per la finestra figure:
  - figure, figure(3), clf, close