Operazioni su Matrici

- Somma e sottrazione
- Operazioni di algebra lineare:
  - Moltiplicazioe
  - Divisione
- Operazioni elemento-per-elemento
- Funzioni di base per matrici
- Matrici di numeri casuali

Addizione/sottrazione di matrici

- Per matrici di identiche dimensioni la somma è definita termine a termine:
  - il comando F = A + B significa

$$F(r,c) = A(r,c) + B(r,c)$$

  - Per ogni coppia riga colonna r,c

Addizione “elemento per elemento”

- Per esempio:

```
A =                B =                >> F = A + B

     1     2            7     8        F =

     3     4            9    10

     5     6           11    12             8    10
                                           12    14
                                           16    18
```

- Note:
  - I vettori devono essere di dimensioni identiche!
  - Un addendo può essere scalare (viene ripetuto)
  - La sottrazione è identica.

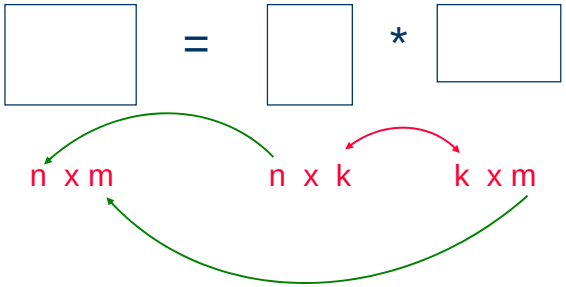
Moltiplicazione di matrici (Algebra Lineare)

- In algebra lineare, l’espressione F = A * B significa

$$F(r,c) = \sum_{k} A(r,k) * B(k,c)$$

  - Ogni elemento è il prodotto scalare di una riga della prima matrice per una colonna della seconda

- Nota bene:
  - Questa operzione solitamente non è commutativa : A*B ≠ B*A
  - Il numero di colonne della 1ª deve coincidere con il numero di righe della 2ª



- Per esempio: qui la moltiplicazione funziona in entrambe le direzioni ma non è commutativa:

```
A =

     1     2
     3     4
     5     6
```

```
B =

     7     9    11
     8    10    12
```

```
>> F = A * B

F =

    23    29    35
    53    67    81
    83   105   127
```

```
>> F = B * A

F =

    89   116
    98   128
```

Ben diverso!

- E qua non funziona per niente:

```
A =

     1     2
     3     4
     5     6
```

```
B =

     7     8
     9    10
    11    12
```

```
>> F = A * B
??? Error using ==> *
Inner matrix dimensions must agree.
```


Applicazioni della moltiplicazione

- Sistema di *n* equazioni in *m* incognite (le *x*)

$$\begin{aligned} a_{11}x_1 &+ a_{12}x_2 \quad \dots \quad + a_{1m}x_m &= b_1 \\ a_{21}x_1 &+ a_{22}x_2 \quad \dots \quad + a_{11}x_m &= b_2 \\ \vdots & \qquad\qquad \ddots \qquad\qquad & \vdots \\ a_{n1}x_1 &+ a_{n2}x_1 \qquad\quad + a_{nm}x_m &= b_n \end{aligned}$$

- Per esempio:

$$\begin{aligned} 2x_1 &+ 3x_2 &+ 3x_3 &= 7 \\ 4x_1 &+ 2x_2 &+ 9x_3 &= 5 \\ 6x_1 &- 7x_2 &+ 2x_3 &= 1 \end{aligned}$$

- In forma matriciale: A * x = b

con

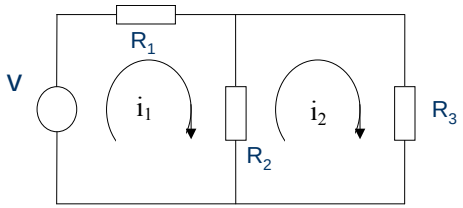
$$A = \begin{bmatrix} 2 & 3 & 3 \\ 4 & 2 & 9 \\ 6 & -7 & 2 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} 7 \\ 5 \\ 1 \end{bmatrix}$$

$$\begin{aligned} a_{11}x_1 &+ a_{12}x_2 &\dots &+ a_{1m}x_m &= b_1 \\ a_{21}x_1 &+ a_{22}x_2 &\dots &+ a_{11}x_m &= b_2 \\ \vdots & & \ddots & & \vdots \\ a_{n1}x_1 &+ a_{n2}x_1 & & + a_{nm}x_m &= b_n \end{aligned}$$

- In generale: A * x = b
  - A è una n per m
  - x è una m per 1
  - b è una n per 1

Vettori colonna (minuscola)

- Utilizzi:
  - Tensioni in statica
  - Flussi in idraulica
  - Flussi di calore
  - es.
  - Corrente nei circuiti
  - Flussi di traffico
  - Economia



$$\begin{bmatrix} R_1 + R_2 & -R_2 \\ R_2 & -(R_2 + R_3) \end{bmatrix} * \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} v \\ 0 \end{bmatrix}$$

Divisione di matrici

- Ricordiamo il comando eye(n)
  - Questa è la matrice identità per la moltiplicazione, I
  - Per ogni matrice A

```
>> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1
```

A * I = I * A = A

Opportunamente dimensionate!

- Poniamo di avere per le matrici quadrate A e B:

$$A * B = B * A = I$$

quindi definiamo l'inversa

$$A = B^{-1} \qquad B = A^{-1}$$

- In Matlab: A ^ -1 o inv(A)
- Quando esiste A^{-1} ?
  - A è quadrata
  - A il determinante non nullo (det(A))

- Per esempio:

```
A =

     0     9     8
     7     4     5
     4     4     2

>> det(A)

ans =

   150
```

```
>> B = A^-1

B =

   -0.0800    0.0933    0.0867
    0.0400   -0.2133    0.3733
    0.0800    0.2400   -0.4200
```

```
>> A*B

ans =

    1.0000         0         0
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000
```

```
>> B*A

ans =

    1.0000    0.0000    0.0000
         0    1.0000   -0.0000
         0    0.0000    1.0000
```


- Per risolvere $A * x = b$
  - Assumiamo che A sia quadrata e det(A) ≠ 0
  - Moltiplichiamo entrambi i membri per A^{-1} a sinistra

$$\underbrace{A^{-1} * A}_{= I} * x = A^{-1} * b$$

$$= x$$

quindi $x = A^{-1} * b$

- In Matlab, x = A \ b o x = inv(A)*b

Barra inversa (backslash)

- Per esempio:

```
A =

     0     9     8
     7     4     5
     4     4     2
```

```
b =

     6
     8
     0
```

```
>> x = A\b

x =

    0.2667
   -1.4667
    2.4000
```

- Verifichiamo:

```
>> A*x

ans =

    6.0000
    8.0000
   -0.0000
```

Operazioni elemento-per-elemento

- Le operazioni elementari (oltre alla somma e differenza) possono essere svolte elmento per elemento usando la notazione puntata (N.B. A e B devono essere della stessa dimensione!)
  - moltiplicazione
    F = A .* B → F(r,c) = A(r,c) * B(r,c)
  - divisione
    F = A ./ B → F(r,c) = A(r,c) / B(r,c)
  - potenza:
    F = A .^ B → F(r,c) = A(r,c) ^ B(r,c)

Il punto!

- Per esempio:

```
a =
     1     2     3

b =
     4     5     6
```

```
>> a.*b

ans =
     4    10    18
```

```
>> a./b

ans =
    0.2500    0.4000    0.5000
```

```
>> a.^b

ans =
     1    32   729
```


- Uno può essere scalare: a = [1 2 3] b = 2

```
>> a.*b

ans =
     2     4     6
```

```
>> b.*a

ans =
     2     4     6
```

```
>> a./b

ans =
    0.5000    1.0000    1.5000
```

```
>> b./a

ans =
    2.0000    1.0000    0.6667
```

```
>> a.^b

ans =
     1     4     9
```

```
>> b.^a

ans =
     2     4     8
```


- Le funzioni predefinite funzionano anche per vettori (elemento per elemento):
  - log , exp
  - Trigonometriche
  - round, ceil, abs
  - etc.

```
>> b = [ 4 9 25; 1 2 10 ]

b =

     4     9    25
     1     2    10

>> sqrt(b)

ans =

    2.0000    3.0000    5.0000
    1.0000    1.4142    3.1623
```

Funzioni per i vettori

- Alcune funzioni predefinite analizzano un vettore e danno come risultato un valore. Per esempio:

```
A =

     6     3     5     1

>> sum(A)

ans =

    15
```

Somma degli elementi

- Altre funzioni per un vettore A:
  - Somma degli elementi: sum(A)
  - Prodotto degli elementi: prod(A)
  - Minimo: min(A)
  - Massimo: max(A)
  - Mediana: median(A)
  - Media: mean(A)
  - Deviazione Standard: std(A)

- Applichiandole ad una matrice, queste funzioni lavorano per colonne!

```
A =

     2     7     5     5
     3     3     3     4
     5     8     7     6

>> sum(A)

ans =

    10    18    15    15
```


- Possiamo però farle lavorare per righe!

```
A =

     2     7     5     5
     3     3     3     4
     5     8     7     6

>> sum(A,2)

ans =

    19
    13
    26
```

questo 2 significa: "usa la 2ª dimensione" cioè lavora per righe!

- Per saperne di più c'è sempre l'help!!!

```
>> help max

 MAX    Largest component.
    For vectors, MAX(X) is the largest element in X. For matrices,
    MAX(X) is a row vector containing the maximum element from each
    column. For N-D arrays, MAX(X) operates along the first
    non-singleton dimension.

    [Y,I] = MAX(X) returns the indices of the maximum values in vector I.
    If the values along the first non-singleton dimension contain more
    than one maximal element, the index of the first one is returned.

    MAX(X,Y) returns an array the same size as X and Y with the
    largest elements taken from X or Y.  Either one can be a scalar.

    [Y,I] = MAX(X,[],DIM) operates along the dimension DIM.
```


- Alcune funzioni producono due valori:
  - min e max possono restituire sia il valore che la sua posizione
  - Di default viene restituito solo il primo risultato

```
A =

     6     3     5     1

>> [value,location] = min(A)

value =

     1


location =

     4
```


- Alcune funzioni danno in uscita un vettore
  - Sort (ordinamento)

```
A =

     6     3     5     1

>> sort(A)

ans =

     1     3     5     6
```


- Oppure vettori multipli:

Valori ordinati

Posizioni originarie

```
A =

     6     3     5     1

>> [vals,locs] = sort(A)

vals =

     1     3     5     6


locs =

     4     2     3     1
```

- Attenzione!!! La funzione sort, se applicata ad una matrice, ordina le colonne indipendentemente!
- Per ordinare mantenendo intatte le righe, si usa la sortrows!
- Questa funzione prende in ingresso la matrice ed il numero di colonna secondo cui ordinarla.

```
a =

    10     5     5     4
     8    10     9     9
     2    10     1     1

>> sort(a)

ans =

     2     5     1     1
     8    10     5     4
    10    10     9     9

>> sortrows(a,1)

ans =

     2    10     1     1
     8    10     9     9
    10     5     5     4
```


Possono tornare utili anche questi comandi:

- flipud(a) : gira la matrice sotto-sopra
- fliplr(a) : gira la matrice destra-sinistra

```
a =

    10     5     5     4
     8    10     9     9
     2    10     1     1

>> flipud(a)

ans =

     2    10     1     1
     8    10     9     9
    10     5     5     4

>> fliplr(a)

ans =

     4     5     5    10
     9     9    10     8
     1     1    10     2
```


Vettori casuali

- Il comando rand è utile per generare vettori di numeri casuali ← pseudo-random

```
>> A = rand(2,3)

A =

    0.6213    0.9568    0.8801
    0.7948    0.5226    0.1730
```

Uniforme in [0,1]

(b-a)*rand(r,c) + a → uniforme in [a, b]

ceil(rand(10,5)*90) → numeri del lotto

- Con randn generiamo numeri casuali con distribuzione normale (media μ=0, varianza σ^2=1)

```
>> B = randn(2,3)

B =

   -0.4326    0.1253   -1.1465
   -1.6656    0.2877    1.1909
```

σ*randn(r,c) + μ → Gaussiana(μ, σ^2)