

# Laboratorio di Informatica T (Ch3)

# Vettori e Matrici in Matlab

---

# Il Perché del “Mat” in Matlab

È ora di fare una precisazione:

- I valori reali, complessi e logici non sono tipi base in Matlab...
- ...Perché il vero tipo base sono le matrici di reali, complessi, logici!

Per Matlab, l'espressione:

10

- Non denota veramente uno scalare...
- ...Ma implicitamente una matrice 1x1
- Lo stesso vale per i valori complessi e logici

Del resto, Matlab sta per “MATrix LABoratory”!

# Matrici in Matlab

Per denotare un matrice si usa la sintassi:

```
[<dato a>, <dato b>, ...; <dato c>, <dato d>, ...; ...]
```

- La “,” è un separatore di colonna
- Il “;” è un separatore di riga

La semantica corrispondente è:

$$\begin{pmatrix} a & b & \dots \\ c & d & \dots \\ \dots & \dots & \dots \end{pmatrix}$$

# Matrici in Matlab

Per esempio, digitando nel prompt dei comandi:

```
[1, 5; 3, 9]
```

Matlab risponde con:

```
ans =  
     1     5  
     3     9
```

- La matrice viene visualizzata formattata per righe e colonne...
- ...almeno finché lo schermo è grande abbastanza

# Matrici e Vettori

Un vettore è semplicemente una matrice  $1 \times n$  o  $n \times 1$ :

```
[1, 2, 3, 4] % un vettore riga  
[1; 2; 3; 4] % un vettore colonna
```

- Notate l'uso di due separatori diversi
- Un vettore riga ha tante colonne (ed una riga sola)
- Un vettore colonna ha tante righe (ed una colonna sola)

Per i vettori riga si può anche usare:

```
[1 2 3 4]
```

- La notazione però è sconsigliata (fa confusione)

# Vettori, Matrici ed Espressioni Composte

La definizione di vettori/matrici è una espressione composta

Quando scrivete...

```
[1+1, 3/4, 4*4]
```

...e battete "invio", Matlab lo interpreta come:

```
[<exp1>, <exp2>, <exp3>]
```

- Le espressioni "<exp1>", "<exp2>", "<exp3>" vengono valutate...
- ...E restituiscono i valori 2, 0.75 e 16
- A questo punto viene costruito il vettore [2, 0.75, 16]

Di fatto, è una chiamata a funzione (con sintassi speciale)!

# Dimensioni di Vettori e Matrici

Si può ottenere il numero di elementi in un vettore con:

```
length(<expr>)    % <expr> deve denotare un vettore
```

Per esempio:

```
length([1, 2, 3]) % Risp: ans = 3  
length(10)       % Risp: ans = 1
```

- Ricordate che gli scalari sono matrici (e quindi anche vettori)!

## Attenzione:

- Evitate di usare la funzione **length** con le matrici
- Non causa errori, ma il comportamento è un po' strano
  - Per dettagli: **help length**



# Dimensioni di Vettori e Matrici

Per ottenere le dimensioni di una matrice potete usare:

```
size(A)
```

- `size` restituisce un vettore, con il numero di righe e colonne:

```
A = [1, 2, 3; 4, 5, 6]    % Corrisponde a [1, 2, 3;
                        %                    4, 5, 6]
B = [1, 2, 3, 4]
C = 10

size(A) % Resp: ans = [2, 3]
size(B) % Resp: ans = [1, 4]
size(C) % Resp: ans = [1, 1]
```

- Ricordate che gli scalari ed i vettori sono matrici!

# Vettori/Matrici ed Operatori

---

# Matrici/Vettori ed Operatori

## Una considerazione interessante:

- Se Matlab utilizza come tipo base le matrici...
- ...Gli operatori aritmetici (“+”, “\*”, etc.) cosa fanno veramente?

## Gli operatori aritmetici in Matlab sono operatori matriciali:

- “+” calcola la somma di due matrici
- “-” calcola la differenza di due matrici
- “\*” calcola il prodotto matriciale (riga per colonna)
- “/” e “^” meritano qualche parola in più

Ma prima, vediamo qualche esempio...

# Matrici/Vettori ed Operatori

```
A = [1, 2; 3, 4] % [1, 2;
                  % 3, 4]
B = [4, 3; 2, 1] % [4, 3;
                  % 2, 1]
C = [2; 4]      % vettore colonna
```

Qualche esempio di applicazione di "+", "-", "\*":

```
A + B % Resp: ans = [5, 5;
                   % 5, 5]
A - B % Resp: ans = [-3, -1;
                   % 1, 3]
A * B % Resp: ans = [ 8, 5;
                   % 20, 13]
A * C % Resp: ans = [10;
                   % 22]
```

# Matrici/Vettori ed Operatori

L'operatore “/” corrisponde alla divisione destra:

- L'espressione  $B / A$  corrisponde a  $BA^{-1}$
- Ossia  $B$ , moltiplicata per l'inversa di  $A$

L'operatore “\” corrisponde alla divisione sinistra

- L'espressione  $A \setminus B$  corrisponde a  $A^{-1}B$
- Ossia l'inversa di  $A$ , moltiplicata per  $B$

L'operatore “^” corrisponde all'esponenziale di matrice:

- Non credo che lo abbiate mai incontrato...
- ...E non credo che lo incontrerete (almeno per quest'anno)

# Matrici/Vettori ed Operatori

Quando uno dei termini è uno scalare:

- L'operatore  $*$  si comporta come in matematica:
- Gli operatori  $/$  e  $\backslash$  anche (in questo caso sono equivalenti!)

```
A = [1, 2; 3, 4]    % Equivale a: [1, 2;
                    %                3, 4]
A * 2              % Denota: [2, 4;
                    %                6, 8]
2 * A             % Denota: [2, 4;
                    %                6, 8]
A / 2             % Denota: [0.5, 1;
                    %                1.5, 4]
2 \ A            % Denota: [0.5, 1;
                    %                1.5, 4]
```

# Matrici/Vettori ed Operatori di Confronto

## Cosa succede per gli operatori di confronto?

- Operano elemento per elemento
- Conseguenza: le due matrici devono avere la stessa dimensione

```
A = [1, 2; 3, 4]    % Equivale a: [1, 2;
                   %                3, 4]
B = [1, 3; 2, 4]    % Equivale a: [1, 3;
                   %                2, 4]

A == B              % Denota: [1, 0;
                   %                0, 1]
A <= B              % Denota: [1, 1;
                   %                0, 1]
```

# Matrici/Vettori e Chiamate a Funzione

## E per quanto riguarda le funzioni?

- La maggior parte delle funzioni predefinite...
- ...Opera sulle matrici elemento per elemento

```
A = [1, 2; 3, 4]    % Equivale a: [1, 2;
                    %                3, 4]
exp(A)             % Denota: [ 2.7183,  7.3891;
                    %                20.0855, 54.5982]
sin(A)             % Denota: [0.8415,  0.9093
                    %                0.1411, -0.7568]
```

**Attenzione:** non vale in tutti i casi!

- Nel dubbio, consultate la documentazione con **help** o **doc**



# Operatori Aritmetici Elemento per Elemento

Un problema che capita spesso:

- Supponiamo di avere due matrici **A** e **B** (o due vettori)...
- ...E di volerle moltiplicare elemento per elemento...

Per questo possiamo usare gli operatori aritmetici "elemento per elemento":

```
A .* B    % Moltiplica gli elementi uno ad uno
A ./ B    % Divide gli elementi uno ad uno
A.^b     % b scalare, eleva a potenza gli elementi
```

- Si chiamano come le loro controparti (i.e. "\*", "/", "^")...
- ...Ma con un "punto" davanti (i.e. ".\*", "./", ".^")

Vedrete che li useremo molto spesso

# Operatore di Trasposizione

Su matrici/vettori si può applicare l'operatore di trasposizione

```
A = [1, 2; 3, 4]    % Equivale a: [1, 2;  
                  %                3, 4]  
  
A.'                % Denota: [1, 3;  
                  %                2, 4]
```

## Attenzione:

- L'operatore di trasposizione è “.’” (con il punto)...
- Perché “'” calcola il complesso coniugato...
- ...Ossia la matrice trasposta, in cui le parti immaginarie sono negate

Nel caso di matrici di numeri reali, i due coincidono

# Costruzione di Matrici/Vettori

---

# Costruzione Rapida di Matrici e Vettori

## Funzioni per Costruire Matrici Notevoli:

- Alcuni tipi di matrice/vettore sono di uso comune...
- ...E Matlab permette di costruirle velocemente

Vediamo qualche esempio rilevante:

```
zeros(N)           % Matrice di zeri, NxN
zeros(M, N)        % Matrice di zeri MxN
ones(N)            % Matrice di uni, NxN
ones(M, N)         % Matrice di uni MxN
eye(N)             % Matrice identità, NxN
eye(M, N)          % Matrice identità estesa, MxN
diag(V)            % Matrice con il vettore V per diagonale
```

# Vettori e Range

Un range costruisce un vettore di elementi consecutivi

La sintassi è:

```
<primo>:<ultimo> % Notazione 1  
<primo>:<incremento>:<ultimo> % Notazione 2
```

Qualche esempio:

- `1:6` equivale a: `[1, 2, 3, 4, 5, 6]`
- `1:2:6` equivale a: `[1, 3, 5]`
- `1:2.5:6` equivale a: `[1, 3.5, 6]`

La costruzione procede finché non si supera il valore `<ultimo>`

# Funzione `linspace`

Una alternativa ai `range` è la funzione `linspace`

Eseguendo la chiamata a funzione:

```
linspace(<primo>, <ultimo>, <numero>)
```

Viene costruito un vettore tale che:

- Il primo elemento è il risultato di `<primo>`
- L'ultimo elemento è il risultato di `<ultimo>`
- Il vettore contiene `<numero>` elementi equispaziati

Per esempio:

```
linspace(0, 1, 5) % Denota [0, 0.25, 0.5, 0.75, 1]
```

# Funzione `linspace`

Il numero di elementi in `linspace` può essere omissivo:

```
linspace(<primo>, <ultimo>)
```

- In questo caso, il vettore conterrà 100 elementi
- Il parametro `<numero>` ha 100 come valore di default

**Tipicamente:**

- Si usa `linspace` quando va bene avere anche numeri reali
- Si usano i range per costruire vettori di numeri interi
- Vedremo un utilizzo importante per i vettori di interi tra poco

# Concatenazione di Vettori/Matrici

Si può costruire una matrice o un vettore per concatenazione:

Basta utilizzarli nella notazione per costruire un nuovo vettore/matrice:

```
A = [ 1, 2 ]
```

```
B = [ 3, 4 ]
```

```
[A, B]      % Denota [1, 2, 3, 4]
```

```
[A; B]      % Denota [1, 2;
```

```
%           3, 4]
```

```
[A', B']    % Denota [1, 3;
```

```
%           2, 4]
```

- Utilizzando “,” si concatena per riga
- Utilizzando “;” si concatena per colonna



# Accesso a *Matrici/Vettori*

---

# Accesso Mediante Indici

Spesso, è utile accedere ad un elemento specifico di un vettore

Ogni elemento di un vettore è associato ad un indice (un intero):

( 1 2 3 4 ... )

- Il primo indice è sempre 1
- L'ultimo indice è uguale al numero di elementi

La notazione:

```
<vettore>(<indice>)
```

Restituisce l'elemento di **<vettore>** in posizione **<indice>**

# Accesso Mediante Indici

## Lo stesso metodo vale per le matrici

In questo caso però si usa un indice doppio:

```
<matrice>(<indice riga>, <indice colonna>)
```

Vediamo qualche esempio:

```
A = [2, 4; 6, 8] % Corrisponde a [2, 4;  
%                               6, 8]  
B = [2, 4, 6]  
  
B(2) % Denota 4  
A(1,1) % Denota 2  
A(2,1) % Denota 6
```

# Accesso Mediante Indici

## Qualche regola sugli indici:

- Devono essere sempre numeri interi
- Devono essere  $\geq 1$
- Devono essere  $\leq$  della lunghezza del vettore/riga/colonna

Per accedere all'ultimo elemento ci sono due modi:

- Usare la lunghezza:

```
V(length(V))
```

- Usare l'indice speciale **end**

```
V(end)
```

# Assegnamento di Elementi

Gli elementi di un vettore/matrice sono assimilabili a variabili

- Quindi il loro valore può essere modificato!
- Si usa l'operatore di assegnamento =

```
A = [1, 2; 3, 4]    % [1, 2;  
                  % 3, 4]  
A(1,2) = 9        % Resp: [1, 9;  
                  %      3, 4]
```

**Valgono le stesse regole delle variabili:**

- Se la matrice (indicizzata) compare a destra dell'='...
  - ...Allora è considerata una espressione e denota un valore
- Se la matrice (indicizzata) compare a sinistra dell'='...
  - ...Allora vi viene assegnato un valore

# Assegnamento ed Estensione

È possibile assegnare elementi “oltre la fine” di un vettore

Supponiamo di avere:

```
v = [1,2,3,5,6,7]
```

Se assegniamo ad un indice  $< 1$ , otteniamo un errore:

```
v(0) = 2 % errore
```

Ma se usiamo un indice  $> \text{length}(x)$ , invece:

```
v(9) = 1 % Otteniamo x = [1,2,3,5,6,7,0,0,1]
```

- Il vettore viene espanso, riempiendo con **0** le celle intermedie

# Assegnamento e Vettore/Matrice Vuoto

Il simbolo `[]` denota un vettore/matrice vuoto

Possiamo definire un vettore vuoto:

```
V = [] % V è vuoto, length(x) è 0
```

Possiamo estendere un vettore vuoto:

```
V(3) = 10 % otteniamo V = [0, 0, 10]
```

Possiamo cancellare un elemento assegnandovi `[]`:

```
x(2) = [] % otteniamo x = [0, 10]
```

L'estensione e `[]` consentono di manipolare la lunghezza

# Indicizzazione Avanzata di Vettori/Matrici

---



# Indicizzazione "Lineare" di Matrici

È possibile accedere ad una matrice con un indice unico

In questo caso, gli elementi si considerano numerati per colonna, e.g.:

$$\begin{pmatrix} \boxed{1} & \boxed{4} & \boxed{7} \\ \boxed{2} & \boxed{5} & \boxed{8} \\ \boxed{3} & \boxed{6} & \boxed{9} \end{pmatrix}$$

Così, per esempio:

```
A = [2, 4; 6, 8] % Corrisponde a [2, 4;
                    %                               6, 8]
A(1) % Denota 2
A(3) % Denota 4
```

- Capita di rado che sia utile...
- ...Ma capita spesso di utilizzarlo per sbaglio :-)

# Indicizzazione Mediante Vettori di Indici

Possiamo accedere ad un sotto-vettore con un vettore di indici:

La sintassi è:

```
<vettore>(<vettore di indici>)
```

Per esempio:

```
V = [2, 4, 6, 8]  
V([2, 3]) % denota [4, 6]
```

- Il risultato è un nuovo vettore
- Contiene gli elementi di **V**, agli indici **2** e **3**

È un metodo particolarmente efficace se si usano i range

# Vettori di Indici: Esempi

Supponiamo di voler sommare le celle adiacenti di:

```
a = [1, 2, 3, 4, 5]
```

- Il risultato deve essere contenuto in un nuovo vettore

# Vettori di Indici: Esempi

Supponiamo di voler sommare le celle adiacenti di:

```
a = [1, 2, 3, 4, 5]
```

- Il risultato deve essere contenuto in un nuovo vettore

Possiamo usare:

```
a(1:end-1) + a(2:end) % Notate l'uso di "end"
```

- `a(1:end-1)` denota [1, 2, 3, 4]
- `a(2:end)` denota [2, 3, 4, 5]

Il risultato è:

```
[3, 5, 7, 9] % [1, 2, 3, 4] + [2, 3, 4, 5]
```

# Vettori di Indici: Esempi

Oppure, supponiamo di voler sommare le celle pari e dispari:

```
a = [1, 2, 3, 4, 5, 6]
```

# Vettori di Indici: Esempi

Oppure, supponiamo di voler sommare le celle pari e dispari:

```
a = [1, 2, 3, 4, 5, 6]
```

Possiamo usare:

```
a(1:2:end) + a(2:2:end)
```

- `a(1:2:end)` denota `[1, 3, 5]`
- `a(2:2:end)` denota `[2, 4, 6]`

Il risultato è:

```
[3, 7, 11]
```

# Vettori di Indici e Matrici

Con le matrici possiamo usare due vettori di indici

La sintassi è:

```
<matrice>(<indici righe>, <indici colonne>)
```

In questo modo viene selezionata una sotto-matrice:

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9] % [1, 2, 3;
                                % 4, 5, 6;
                                % 7, 8, 9]
A(2:3, 1:2) % Ultime due righe, prime due colonne
            % Denota [4, 5;
            %          7, 8]
```

# Vettori di Indici e Matrici

Uno dei due indici può essere non specificato, con “:”

Per esempio, data:

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

Possiamo selezionare le prime due righe:

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

O la seconda colonna:

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



# Indicizzazione Mediante Valori Logici

Infine, possiamo indicizzare mediante valori logici:

```
<vettore/matrice A>( <vettore/matrice B> )
```

- Restituisce come sotto-vettore/sotto-matrice...
- ...Gli elementi di **A** in corrispondenza dei quali **B** contiene **true**

I due vettori/matrici devono avere la stessa dimensione

```
A = [1, 2; 3, 4] % [1, 2;  
                % 3, 4]  
A([true, true; false, false]) % Denota [1, 2]
```

- Funziona solo se il vettore/matrice B contiene valori logici

# Indicizzazione Mediante Valori Logici

Vediamo un utilizzo tipico:

```
V = [2, 6, 4, 7]
B = (V < 5)    % Denota [true, false, true, false]
V(B)          % Denota [2, 4]
```

- In questo modo otteniamo gli elementi di **V** minori di 5

Si può anche evitare di usare la variabile **B**

```
V(V < 5)
```

- Come al solito, prima viene valutato **V < 5**...
- ...E poi viene effettuata l'indicizzazione

# Assegnamento di Vettori/Matrici

È possibile assegnare in un solo colpo un sotto-vettore/matrice:

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

A(2:end, 2:end) = [1, 0; 0, 1] % Risp. [1, 2, 3;
                                         % 4, 1, 0,
                                         % 7, 0, 1]
```

- È bene che gli elementi a sx e dx dell'operatore "=" ...
- ...abbiamo la stessa dimensione...
- In caso contrario, Matlab cerca di adattarle
- È un comportamento voluto e si chiama broadcasting

# Broadcasting

Il broadcasting è un meccanismo che:

- Con alcuni operatori (e.g. assegnamento e “.\*”)...
- ...Permette a Matlab di modificare le dimensioni di matrici...
- ...Che risulterebbero altrimenti incompatibili

Per esempio:

```
[1; 2] .* [1, 3] % colonna per riga
```

Viene espanso come (replica di righe/colonne):

$$\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix} .* \begin{pmatrix} 1 & 3 \\ 1 & 3 \end{pmatrix}$$

# Broadcasting

## Il broadcasting richiede esperienza per essere utilizzato

- Noi non lo useremo mai (o quasi)
- Però è bene sapere che esiste!
- Infatti potreste innescarlo accidentalmente...
- ...Semplicemente usando matrici/vettore con dimensioni “sbagliate”

Il “quasi” di cui sopra si riferisce ad un caso particolare:

### Assegnamento in blocco di uno scalare:

```
A(:,1) = 7    % Risp: [7, 9;  
                %      7, 4]
```

- In questo caso, il valore viene inserito in tutti gli elementi indicizzati

**File di Script**

# File di Script

In Matlab, si chiama file di script:

- Un file di testo con l'estensione **.m**
- ...Che contiene una sequenza di istruzioni

In sostanza, è un file con un programma scritto in Matlab

**Un file di script può essere eseguito con la sintassi:**

```
<nome del file senza ".m"> + [INVIO]
```

- Per esempio **zeta.m** si esegue con **zeta** + [INVIO]
- Il file deve essere nella cartella corrente
- Il file deve essere nella cartella corrente!!!

Eseguire uno script equivale a scrivere le sue istruzioni sul prompt

# Qualche Dritta sugli Script

## Scegliete il nome con un filo di attenzione

- Se chiamate lo script `pi.m...`
- ...quando scrivete l'istruzione `pi+[INVIO]...`
- ...eseguirete lo script, invece di ottenere il valore di  $\pi$ !

Soluzione:

- Se il nome che volete esiste già, modificatelo leggermente...
- ...Per esempio aggiungendo un prefisso/suffisso
- E.g. `es1.m`  $\longrightarrow$  `ch4_es1.m`



# Qualche Dritta sugli Script

**Mantenete il codice leggibile**, in particolare:

- Usate l'indentazione
- Usate l'indentazione!!!
- Non comprimete troppe operazioni in una sola riga

```
sum(all(a(1:2:end, :)(:) * b')) % UNA BRUTTA IDEA
```

- Tra due parentesi, potete andare a capo liberamente:

```
a = [1, 2, 3, 4, 5, 6, 7, 8,  
     9, 10, 11, 12, 13, 14]
```

- Se non ci sono parentesi, potete andare a capo con con "..."

# Qualche Dritta sugli Script

È possibile inserire commenti nel codice:

Li abbiamo già visti, la sintassi è:

```
% Ecco un commento!
```

- Il testo che segue il simbolo “%” viene ignorato

Commentare è importante: aiuta a ragionare e ricordare

- Non ci credete?
- Un giorno un programmatore trovò questa cosa nel suo codice :-)

```
% When I wrote this, only God and I knew  
%     what I was doing  
% Now, God only knows
```

**Un Po' di Esercizi**

---

# Inserimento di Vettori Matrici

Dal sito del corso, scaricate lo “start-kit” per questa lezione

- Estraiete il contenuto dell'archivio `.zip` in una cartella
- **NOTA:** Non usate il doppio click!
  - Se lo fate, Windows permette di esplorare l'archivio `.zip`...
  - ...Ma non procede all'estrazione
  - Risultato: Matlab non potrà salvare le vostre modifiche
- In Matlab “spostatevi” nella cartella appena creata
  - Usate l'icona vicino alla barra degli indirizzi

Lo script assegna alla variable **VV** il vettore:

$$(2 \quad 4 \quad 6 \quad 8)$$

...Ed alla variable **AA** la matrice:

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \\ 25 & 27 & 29 & 31 \end{pmatrix}$$

# Accesso a Sotto-Vettori

Nello script, provate ad accedere ai seguenti sotto-vettori di **vv**:

( 2 4 6 8 )

- Il vettore (2, 8) con il primo e l'ultimo elemento
- Il vettore (4, 6) con i due elementi in mezzo
- Il vettore (2, 6) con gli elementi ad indici dispari
- Il vettore (4, 8) con gli elementi ad indici pari
- Il vettore con gli elementi minori del valore 5
- Il vettore con gli elementi  $< 8$  e  $> 2$  (usate le condizioni)
- Il vettore ( 8 6 4 2 ) con gli elementi in ordine inverso

All'occorrenza, utilizzate le modalità di indicizzazione avanzata!

# Accesso a Sotto-Matrici

Nello script, provate ad accedere alle seguenti sotto-matrici di **AA**:

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \\ 25 & 27 & 29 & 31 \end{pmatrix}$$

- La seconda colonna
- La seconda riga
- La matrice con gli elementi al centro: **[11, 13; 19, 21]**
- La matrice corrispondente ad **A**, senza la seconda riga e colonna

# Calcolo Matriciale/Vettoriale

Calcolate il risultato dell'espressione lineare:

$$Ab + c$$

Dove:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad c = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

- Le matrici ed i vettori necessari sono già forniti nello start-kit

# Calcolo Matriciale/Vettoriale

## Risoluzione di sistemi di equazioni lineari

- Dovreste sapere che un sistema di equazioni lineari...
- ...Può essere espresso in forma matriciale:

$$Ax = b$$

- La soluzione è quindi:  $x = A^{-1}b$  (se  $A$  è non singolare)

## Consideriamo il sistema:

$$x_1 + x_2 + x_3 = 2$$

$$x_1 + 2x_2 = 1$$

$$2x_1 + x_3 = 0$$

- Definire la matrice  $A$  dei coefficienti
- Definire il vettore  $b$  dei termini noti
- Calcolare la soluzione



# Valutazione di Espressioni Vettoriali

Valutate le seguenti espressioni, per  $x = (0, 1, 2, 3, 4, 5)$

- $2x^2 - 3x + 1$
- $x \left( \frac{1}{x} + 1 \right)$
- $\frac{1}{1+e^{-x}}$
- $\sqrt{1+x} - \log(1+x)$

Procedete in questo modo:

- Prima costruite un vettore con i valori di  $x$
- Poi valutate le espressioni utilizzando gli [operatori punto a punto](#)

```
x = 0:3
x.^2          % Un esempio, per x^2 con x = 0,1,2,3
```

# Costruzione di Vettori e Matrici

Costruite, senza immettere direttamente i valori:

- Il vettore (1, 2, 3, 4, 5)
- Il vettore (1, 4, 7, 10, 13)
- Il vettore (1, 2, 3, 11, 12, 13)
- La matrice:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

Dovrete usare delle espressioni (o chiamate a funzione)

# Assegnamento di Matrici e Vettori

Costruite, senza immettere i valori uno per uno:

- Una matrice  $\mathbf{C}$  identica  $\mathbf{A}$ , ma avente  $\mathbf{v}$  come prima riga
- Un vettore  $\mathbf{z}$  che contenga  $(0, 0, 0, 0, 0, 0)$
- Il vettore  $\mathbf{z}$  modificato in modo che contenga  $(0, 1, 0, 1, 0, 1)$
- Una matrice  $\mathbf{T}$  così fatta:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$