

 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

MATLAB, from zero to ~~hero~~ structs

Matteo Ferroni
matteo.ferroni@polimi.it

16/12/2016



POLITECNICO
DI MILANO



MATLAB, from zero to ~~hero~~

zero to hero

structs

the definition is split in to two parts

zero: being a loser, someone with no aspirations or goals

hero: being amazing person who does what they please and makes people want to be them; the ultimate in someone being an all around great person

Wow that boy went from zero to hero in a matter of minutes!

Da tenere sempre a mente...

In MATLAB si ragiona con un paradigma diverso dal C:

- tutto è un array;
- gli **scalari** sono un caso particolare di array multidimensionale;
- le funzioni operano **in parallelo sui dati**, contenuti in forma array multidimensionale nelle variabili;
- le operazioni matriciali sono esprimibili con **poche istruzioni** (al limite una sola istruzione)
- i **for** sono quasi sempre da **evitare**
- le maschere di bit tornano molto comode quando dovete **filtrare** in qualche modo i dati;

Riguardo i cicli: vettorizzazione (1)

- In molti casi è possibile sostituire un for con l'uso di un opportuno vettore. Esempio

```
%calcolo del quadrato degli interi tra 1 e 100  
for ii=1:100  
    square(ii)=ii^2;  
end
```

Riguardo i cicli: vettorizzazione (1)

- In molti casi è possibile sostituire un for con l'uso di un opportuno vettore. Esempio

```
%calcolo del quadrato degli interi tra 1 e 100  
for ii=1:100  
    square(ii)=ii^2;  
end
```

%frammento di codice equivalente: vettorizzazione

```
ii=1:100;  
square=ii.^2;
```

NB: bisogna usare la versione `'.'` che opera elemento per elemento

Riguardo i cicli: vettorizzazione (1)

- In molti casi è possibile sostituire un for con l'uso di un opportuno vettore. Esempio

```
%calcolo del quadrato degli interi tra 1 e 100  
for ii=1:100  
    square(ii)=ii^2;  
end
```

```
%frammento di codice equivalente: vettorizzazione
```

```
ii=1:100;  
square=ii.^2;
```

NB: bisogna usare la versione `).^` che opera elemento per elemento

- **La versione con il for può essere fino a 15 volte più lenta della versione con la vettorizzazione!**

Riguardo i cicli: vettorizzazione (2)

- Riprendiamo l'esempio
 - $b = a > 5$
 - $\text{sqrt}(a(b))$
 - $a(b) = \text{sqrt}(a(b))$

Riguardo i cicli: vettorizzazione (2)

- Riprendiamo l'esempio

- $b = a > 5$
- $\text{sqrt}(a(b))$
- $a(b) = \text{sqrt}(a(b))$

- Esecuzione dello stesso calcolo con i cicli

`[r, c]=size(a); %usata in questo modo size dà righe e colonne di una matrice`

`for h = 1:r`

`for k = 1:c`

`if a(h, k) > 5`

`a(h, k) = sqrt(a(h, k));`

`end`

`end`

`end`

Riguardo i cicli: vettorizzazione (2)

- Riprendiamo l'esempio
 - $b = a > 5$
 - $\text{sqrt}(a(b))$
 - $a(b) = \text{sqrt}(a(b))$
- Esecuzione dello stesso calcolo con i cicli

```
[r, c]=size(a); %usata in questo modo size dà righe e colonne di una matrice
for h = 1:r
    for k = 1:c
        if a(h, k)>5
            a(h, k)=sqrt(a(h, k));
        end
    end
end
end
```
- Anche qui il codice che sfrutta la vettorizzazione è molto più efficiente dell'altro

Agenda

(10') Es1 - Vettori e funzioni base

(20') Es1.2 - Vettori, vettori everywhere!

(15') Es2 - Stringhe palindrome

(30') Es3 - Files e matrici

(30') Es4 - Maschere di bit (ordinamento v1.0)

(20') Es7 - Struct (film)

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

Scrivere un programma che permetta all'utente di inserire un vettore di numeri interi. Dopo aver verificato che l'array inserito sia numerico, effettuare i seguenti controlli:

- Verificare se tutti i numeri sono positivi
- Verificare se esiste un numero negativo
- Applicare la radice quadrata a tutti i valori: ci sono dei valori complessi?
- Verificare se tutti i numeri sono pari e trovarne le posizioni
- Verificare se esiste un numero dispari
- Contare i numeri dispari se esistono e dire in che posizione sono

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Scrivere un programma che permetta all'utente di  
% inserire un vettore di numeri interi.  
  
% Dopo aver verificato che l'array inserito sia numerico,  
%effettuare i seguenti controlli:
```

Esercizio: Vettori e funzioni base

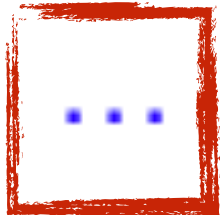
Scorsa
esercitazione

```
% Scrivere un programma che permetta all'utente di  
% inserire un vettore di numeri interi.  
vett = input('Inserisci un vettore: ');  
  
% Dopo aver verificato che l'array inserito sia numerico,  
%effettuare i seguenti controlli:
```

Esercizio: Vettori e funzioni base

```
% Scrivere un programma che permetta all'utente di
% inserire un vettore di numeri interi.
vett = input('Inserisci un vettore: ');

% Dopo aver verificato che l'array inserito sia numerico,
%effettuare i seguenti controlli:
if isnumeric(vett)
    ...
else
    disp('Devi inserire un array numerico');
end
```



Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Verificare se tutti i numeri sono positivi
```

Esercizio: Vettori e funzioni base

```
% Verificare se tutti i numeri sono positivi
if all(vett > 0)
    disp('tutti i numeri sono positivi');
else
    disp('non tutti i numeri sono positivi');
end

% Verificare se esiste un numero negativo
```


Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Verificare se tutti i numeri sono positivi
if all(vett > 0)
    disp('tutti i numeri sono positivi');
else
    disp('non tutti i numeri sono positivi');
end

% Verificare se esiste un numero negativo
if any(vett < 0)
    disp('esiste un numero negativo');
else
    disp('non esiste alcun numero negativo');
end
```

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Applicare la radice quadrata a tutti i valori:  
% ci sono dei valori complessi?
```

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Applicare la radice quadrata a tutti i valori:  
% ci sono dei valori complessi?  
sqrt_vett = sqrt(vett)
```

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Applicare la radice quadrata a tutti i valori:  
% ci sono dei valori complessi?  
sqrt_vett = sqrt(vett)  
if isreal(sqrt_vett)  
    disp('non esistono valori complessi');  
else  
    disp('esistono valori complessi');  
end  
  
% Verificare se tutti i numeri sono pari e  
% trovarne le posizioni
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:  
% ci sono dei valori complessi?  
sqrt_vett = sqrt(vett)  
if isreal(sqrt_vett)  
    disp('non esistono valori complessi');  
else  
    disp('esistono valori complessi');  
end  
  
% Verificare se tutti i numeri sono pari e  
% trovarne le posizioni  
vett_mod = mod(vett,2);
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:  
% ci sono dei valori complessi?  
sqrt_vett = sqrt(vett)  
if isreal(sqrt_vett)  
    disp('non esistono valori complessi');  
else  
    disp('esistono valori complessi');  
end  
  
% Verificare se tutti i numeri sono pari e  
% trovarne le posizioni  
vett_mod = mod(vett,2);  
if(all(vett_mod==0))  
    disp('tutti i numeri sono pari');  
else  
    disp('non tutti i numeri sono pari');  
end
```

Esercizio: Vettori e funzioni base

```
% Applicare la radice quadrata a tutti i valori:
% ci sono dei valori complessi?
sqrt_vett = sqrt(vett)
if isreal(sqrt_vett)
    disp('non esistono valori complessi');
else
    disp('esistono valori complessi');
end

% Verificare se tutti i numeri sono pari e
% trovarne le posizioni
vett_mod = mod(vett,2);
if(all(vett_mod==0))
    disp('tutti i numeri sono pari');
else
    disp('non tutti i numeri sono pari');
end
pos_pari = find(1-vett_mod);
disp('I numeri pari sono in posizione: ');
disp(pos_pari);
```


Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Verificare se esiste un numero dispari
```


Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Verificare se esiste un numero dispari
if any(vett_mod),
    disp('esiste almeno un numero dispari');

    % Contare i numeri dispari se esistono e dire in che posizioni sono

else
    disp('non esistono numeri dispari');
end
```

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Verificare se esiste un numero dispari
if any(vett_mod),
    disp('esiste almeno un numero dispari');

    % Contare i numeri dispari se esistono e dire in che posizioni sono
    count_dispari = sum(vett_mod);
    disp(['i numeri dispari sono ' num2str(count_dispari)]);

else
    disp('non esistono numeri dispari');
end
```

Esercizio: Vettori e funzioni base

Scorsa
esercitazione

```
% Verificare se esiste un numero dispari
if any(vett_mod),
    disp('esiste almeno un numero dispari');

    % Contare i numeri dispari se esistono e dire in che posizioni sono
    count_dispari = sum(vett_mod);
    disp(['i numeri dispari sono ' num2str(count_dispari)]);
    pos_dispari = find(vett_mod);
    disp('i numeri dispari sono in posizione: ');
    disp(pos_dispari);
else
    disp('non esistono numeri dispari');
end
```

Agenda

~~(10') Es1 - Vettori e funzioni base~~

(20') Es1.2 - Vettori, vettori everywhere!

(15') Es2 - Stringhe palindrome

(30') Es3 - Files e matrici

(30') Es4 - Maschere di bit (ordinamento v1.0)

(20') Es7 - Struct (film)

Esercizio: Vettori, vettori everywhere!

- Scrivere un programma che letti da tastiera due vettori numerici scambi gli elementi di indice pari del primo vettore con quelli di indice dispari del secondo.

NOTA: Si continui a chiedere all'utente di inserire dei vettori fino a quando questi non soddisfano le condizioni necessarie per risolvere l'esercizio.

Esercizio: Vettori, vettori everywhere!

```
v1(2:2:end) = v2(1:2:end);
```

Esercizio: Vettori, vettori everywhere!

```
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) =
```

Esercizio: Vettori, vettori everywhere!

```
temp = v1(2:2:end);  
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) = temp;
```


Esercizio: Vettori, vettori everywhere!

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo vettore (tra []) : ');
```

end

```
temp = v1(2:2:end);  
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) = temp;
```

v1
v2

Esercizio: Vettori, vettori everywhere!

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
```

end

```
temp = v1(2:2:end);  
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) = temp;
```

v1
v2

Esercizio: Vettori, vettori everywhere!

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;  
vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
```

end

```
temp = v1(2:2:end);  
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) = temp;
```

v1
v2

Esercizio: Vettori, vettori everywhere!

while

```
v1 = input('Inserisci il primo vettore (tra []) : ');  
v2 = input('Inserisci il secondo vettore (tra []) : ');
```

```
pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;  
vettore = size(v1, 1) == 1 & size(v2, 1) == 1;  
numerico = isnumeric(v1) & isnumeric(v2);
```

end

```
temp = v1(2:2:end);  
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) = temp;
```

v1
v2

Esercizio: Vettori, vettori everywhere!

```
while(pari == false || vettore == false || numerico == false)
    v1 = input('Inserisci il primo vettore (tra []) : ');
    v2 = input('Inserisci il secondo vettore (tra []) : ');

    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;
    numerico = isnumeric(v1) & isnumeric(v2);
end

temp = v1(2:2:end);
v1(2:2:end) = v2(1:2:end);
v2(1:2:end) = temp;
```

v1

v2

Esercizio: Vettori, vettori everywhere!

```
pari = false;  
vettore = false;  
numerico == false
```

```
while(pari == false || vettore == false || numerico == false)  
    v1 = input('Inserisci il primo vettore (tra []) : ');  
    v2 = input('Inserisci il secondo vettore (tra []) : ');  
  
    pari = mod(size(v1, 2), 2) == 0 & mod(size(v2, 2), 2) == 0;  
    vettore = size(v1, 1) == 1 & size(v2, 1) == 1;  
    numerico = isnumeric(v1) & isnumeric(v2);  
end
```

```
temp = v1(2:2:end);  
v1(2:2:end) = v2(1:2:end);  
v2(1:2:end) = temp;
```

```
v1  
v2
```

Agenda

~~(10') Es1 - Vettori e funzioni base~~

~~(20') Es1.2 - Vettori, vettori everywhere!~~

(15') Es2 - Stringhe palindrome

(30') Es3 - Files e matrici

(30') Es4 - Maschere di bit (ordinamento v1.0)

(20') Es7 - Struct (film)

Esercizio: Stringhe palindrome

Scrivere uno script riceva in input una stringa e dica se è *palindroma*.

Esercizio: Stringhe palindrome

```
stringa = input('Inserire una stringa: ');
```

Esercizio: Stringhe palindrome

```
stringa = input('Inserire una stringa: ');  
  
% Ricavo la stringa invertita  
stringa_r = stringa(end:-1:1);
```

Esercizio: Stringhe palindrome

```
stringa = input('Inserire una stringa: ');

% Ricavo la stringa invertita
stringa_r = stringa(end:-1:1);

% Se le stringhe sono uguali, la stringa palindroma
if all(stringa == stringa_r)
    disp('La stringa è palindroma!');
else
    disp('La stringa NON è palindroma!')
end
```

Agenda

~~(10') Es1 - Vettori e funzioni base~~

~~(20') Es1.2 - Vettori, vettori everywhere!~~

~~(15') Es2 - Stringhe palindrome~~

(30') Es3 - Files e matrici

(30') Es4 - Maschere di bit (ordinamento v1.0)

(20') Es7 - Struct (film)

Esercizio: Files e matrici

La **variabile prezzi** contiene le informazioni riguardanti i prezzi della benzina per una serie di compagnie nel mese di Settembre. Il **file** contiene una **matrice prezzi NxM** dove N indica il *giorno* del mese in cui è stato registrato il prezzo, mentre M è l'indice che identifica la *compagnia*. Il valore `prezzi(4,3)` conterra' quindi il costo della benzina per il giorno 4 presso la compagnia 3.

- Scrivere un programma Matlab che carichi la variabile prezzi contenuta in un file di nome 'prezzi.mat'.
- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese:
 - Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
 - Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?
 - Calcolare quanto è variato nel corso del mese il prezzo praticato dalle compagnie
 - Qual è la compagnia che durante il mese ha aumentato maggiormente il prezzo, e di quanto?
- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

Esercizio: Files e matrici

```
% Carico il file che contiene la definizione dell'array prezzi  
load 'prezzi'
```

- Trovare il vettore che contenga i prezzi praticati dalle compagnie durante il primo giorno del mese

Esercizio: Files e matrici

```
% Carico il file che contiene la definizione dell'array prezzi  
load 'prezzi'
```

```
prezzi1 = prezzi(1,:)
```

- Qual è stato il prezzo massimo e minimo, per ciascuna compagnia, praticato durante il mese?
- Qual è stato, per ciascun giorno del mese, il prezzo massimo (e minimo) a cui trovare la benzina?

Esercizio: Files e matrici

```
% Carico il file che contiene la definizione dell'array prezzi  
load 'prezzi'
```

```
prezzi1 = prezzi(1,:)
```

```
prezziBrandMin = min(prezzi)  
prezziBrandMax = max(prezzi)
```

```
prezziGiornoMin = min(prezzi, [], 2)  
prezziGiornoMax = max(prezzi, [], 2)
```

- **min(X,[],DIM)** operates along the dimension **DIM**

```
% Calcolo la variazione del prezzo come la differenza tra  
% i prezzi minimi e massimi nel mese
```

```
% Massima variazione
```

```
% Compagnia con massima variazione
```


Esercizio: Files e matrici

```
% Carico il file che contiene la definizione dell'array prezzi  
load 'prezzi'
```

```
prezzi1 = prezzi(1,:)
```

```
prezziBrandMin = min(prezzi)  
prezziBrandMax = max(prezzi)
```

```
prezziGiornoMin = min(prezzi, [], 2)  
prezziGiornoMax = max(prezzi, [], 2)
```

```
% Calcolo la variazione del prezzo come la differenza tra  
% i prezzi minimi e massimi nel mese  
diffBrand = prezziBrandMax - prezziBrandMin
```

```
% Massima variazione
```

```
% Compagnia con massima variazione
```

Esercizio: Files e matrici

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)

prezziBrandMin = min(prezzi)
prezziBrandMax = max(prezzi)

prezziGiornoMin = min(prezzi, [], 2)
prezziGiornoMax = max(prezzi, [], 2)

% Calcolo la variazione del prezzo come la differenza tra
% i prezzi minimi e massimi nel mese
diffBrand = prezziBrandMax - prezziBrandMin

% Massima variazione
maxDiff = max(diffBrand)

% Compagnia con massima variazione
```

Esercizio: Files e matrici

```
% Carico il file che contiene la definizione dell'array prezzi
load 'prezzi'

prezzi1 = prezzi(1,:)

prezziBrandMin = min(prezzi)
prezziBrandMax = max(prezzi)

prezziGiornoMin = min(prezzi, [], 2)
prezziGiornoMax = max(prezzi, [], 2)

% Calcolo la variazione del prezzo come la differenza tra
% i prezzi minimi e massimi nel mese
diffBrand = prezziBrandMax - prezziBrandMin

% Massima variazione
maxDiff = max(diffBrand)

% Compagnia con massima variazione
maxBrand = find(diffBrand == maxDiff)
```

Esercizio: Files e matrici

- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

```
g = input('inserire il giorno ');
```

Esercizio: Files e matrici

- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

```
g = input('inserire il giorno ');
```

```
%trova il minimo prezzo del giorno g  
minp = min(prezzi(g,:));
```

```
%trova la compagnia/compagnie con prezzo minimo  
comp = find(prezzi(g,)==minp);
```

Esercizio: Files e matrici

- Dato un giorno del mese, si ricavi la compagnia/le compagnie che erano più convenienti in quello specifico giorno, e il prezzo.

```
g = input('inserire il giorno ');

%trova il minimo prezzo del giorno g
minp = min(prezzi(g,:));

%trova la compagnia/compagnie con prezzo minimo
comp = find(prezzi(g,)==minp);

disp(['La più conveniente il giorno ' num2str(g)
      ' era/erano la compagnia/compagnie ' mat2str(comp)
      ' al prezzo di ' num2str(minp)])
```

Agenda

~~(10') Es1 - Vettori e funzioni base~~

~~(20') Es1.2 - Vettori, vettori everywhere!~~

~~(15') Es2 - Stringhe palindrome~~

~~(30') Es3 - Files e matrici~~

(30') Es4 - Maschere di bit (ordinamento v1.0)

(20') Es7 - Struct (film)

Esercizio: Maschere di bit (ordinamento v1)

- Si ordini un array di n elementi facendo uso delle istruzioni messe a disposizione da matlab.

Algoritmo di ordinamento: dato un array “disordinato” ed un array “ordinato”, trova il valore minimo nell’array “disordinato” e mettilo nella prima posizione libera dell’array “ordinato”, quindi rimuovilo dall’array “disordinato”. Itera fino a che “disordinato” é lungo 0.