

 POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Settimana esercitazione

Riccardo Cattaneo

Alessandro A. Nacci



POLITECNICO
DI MILANO

13/11/2013



Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base²
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Blocco 3 / 1 (10')

```
% Dato un vettore in ingresso ed un numero a (entrambi float)
% controllare quante occorrenze del numero sono presenti
% nel vettore.
% In seguito elencare i numeri diversi da a presenti nel vettore
```

Blocco 3 / 1 (10')

```
vett = input('Inserisci un vettore ')\na = input('inserisci un numero con cui effettuare il confronto: ')\n%%\n%% Ricerca dei numeri uguali\n%% SONO NUMERI IN VIRGOLA MOBILE! Usare l'operatore '==' può\n%% portare ad errori dovuti alle approssimazioni!\nfprintf("\n\n\nIl numero di occorrenze del numero %f sono ", a)\ndisp(sum(abs(vett - a) <= eps))\nfprintf("I numeri diversi da %f sono:\n", a)\ndisp(vett(vett ~= a))\n%%\n%% Ricerca dei numeri maggiori\nfprintf("\nI numeri maggiori di %f sono ", a)\ndisp(sum(vett > a))\nfprintf("e in particolare sono:\n")\ndisp(vett(vett > a))
```

Blocco 3 / 2 (15')

```
% Se l'array inserito è 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
% Trovare gli indici dei numeri pari
% Verificare se esiste un numero dispari
    % Contare i numeri dispari se esistono
    % Dire in che posizioni sono
```

Blocco 3 / 2 (15')

```
vett = input('Inserisci un vettore ')
if isnumeric(vett)
    %numeri positivi
    if all(vett > 0),
        disp('tutti i numeri sono positivi')
    else
        disp('non tutti i numeri sono positivi')
    end

    %esistenza numero negativo
    if any(vett < 0),
        disp('esiste un numero negativo')
    else
        disp('non esiste alcun numero negativo')
    end

    %radice quadrata
    sqrt_vett = sqrt(vett)
    if isreal(sqrt_vett),
        disp('non esistono valori complessi')
        disp('esistono valori complessi')
    end
end
```

Blocco 3 / 2 (15')

```
%numeri pari
carry_vett = mod(vett,2)
if(all(carry_vett==0))
    disp('tutti i numeri sono pari')
else
    disp('non tutti i numeri sono pari')
end
%indici dei numeri pari
pos_pari = find(1-carry_vett)
%numeri dispari
if any(carry_vett)
    disp('esiste almeno un numero dispari')
    pos_dispari = find(carry_vett),
    disp('i numeri dispari sono in posizione')

    disp(pos_dispari)
else
    disp('non esistono numeri dispari')
end
else % dell'"if isnumeric(...)"
    disp('Devi inserire un array numerico')
end
```

Blocco 3 / 3 (10')

```
% DISEGNARE LA TAVOLA PITAGORICA di dimensione nxn
```

```
%
```

```
% 1 2 3 4 5
```

```
% 2 4 6 8 10
```

```
% 3 6 9 12 15
```

```
% 4 8 12 16 20
```

```
% 5 10 15 20 25 %
```

```
%
```

```
% Per risolvere quest'esercizio, è sufficiente osservare la struttura della  
% tavola pitagorica: la prima riga è un vettore di numeri naturali, da 1 a  
% 10, così come la prima colonna; mentre l'elemento (i,j)-esimo è ottenuto  
% moltiplicando l'elemento j-esimo della prima riga e i-esimo della prima  
% colonna.
```


Blocco 3 / 3 (10')

```
% Ecco che è possibile vedere la colonna come la trasposta della prima riga,  
% e gli altri valori sono ottenibili effettuando una moltiplicazione tra  
% vettori  
n = input('Inserire la dimensione della tavola pitagorica')  
%Determino la riga della tavola pitagorica  
a=1:n  
%Calcolo (e visualizzazione) della tavola periodica  
a'*a
```

Blocco 3 / 4 (10')

```
% Progettare uno script matlab che chieda all'utente quale tra le  
% seguenti operazioni si vuole effettuare su un generico vettore  
%     MEDIA MASSIMO MINIMO
```

Blocco 3 / 4 (10')

```
vettore = []
```

%Creazione menù: per prima cosa occorre che l'utente sia a conoscenza dell'esistenza di un menù e delle opzioni di scelta; per far ciò è sufficiente visualizzare delle stringe sul command window che indichino le opzioni disponibili

```
disp('selezionare un azione');  
disp('1: calcolo della media');  
disp('2: calcolo del massimo');  
disp('3: calcolo del minimo');  
disp('4: esci');
```

```
scelta=input('scelta: ');
```

%Le righe di codice precedenti hanno la SOLA funzione di visualizzare il menù. Per far sì che sia realmente un menu, occorre valutare con un test il contenuto della scelta effettuata (variabile scelta) A tal scopo, l'uso della struttura switch/case è consigliabile, in quando si hanno un numero di opzioni finite e note a priori

```
switch scelta  
    case 1  
        out=mean(vettore);  
    case 2  
        out=max(vettore);  
    case 3  
        out=min(vettore);  
    case 4  
end
```

%in questo caso, uscire corrisponde a non eseguire nessuna operazione

```
disp('esco')
```

Blocco 3 / 5 (15')

```
% Quanto guadagna ogni operaio?
% Qual è il salario totale di tutti gli operai?
% Quanti pezzi vengono prodotti?
% Qual è il costo medio di un pezzo?
% Quante ore occorrono in media per un pezzo?
% Qual è l'operaio più efficiente?

% Dati iniziali (operaio i-esimo, posizione i-esima)
pagaoraria=[5 5.5 6.5 5 6.25]
oresett=[40 43 37 50 45]
pezzi=[1000 1100 1000 1200 1100]
```

Blocco 3 / 5 (15')

```
% quanto guadagna ogni operaio?
% per rispondere a questa domanda, è sufficiente effettuare una
% moltiplicazione elemento per elemento di vettori paga oraria e oresett
paghe=pagaoraria.*oresett
% Qual'è il salario totale di tutti gli operai?
% Si chiede in pratica di sommare gli elementi della riga 'paghe', che per
% noi è un vettore. è quindi possibile usare direttamente la funzione
' sum',
% che effettua la somma di tutti gli elementi nel vettore in argomento
salariototale=sum(paghe)
% Quanti pezzi vengono prodotti?
% In analogia a quanto detto, si riutilizza la funzione 'sum'
pezzitotale=sum(pezzi)
```

Blocco 3 / 5 (15')

```
% Qual'è il costo medio di un prezzo?  
% Il costo medio si ricava considerando il numero di pezzi prodotti in  
totale  
% ed il costo totale di produzione (costo operai). Al pezzo, quindi si ha:  
% costo pezzo= salariototale/pezzitotale, valori già calcolati con le  
% domande precedenti  
costopezzo=salariototale/pezzitotale  
% Quante ore occorrono in media per pezzo?  
% Considerazioni analoghe al precedente, con la differenza che occorre  
% conoscere le ore totali, fin'ora non calcolate, valore ottenibile  
% facilmente con la funzione 'sum'  
oreperpezzo=sum(oresett)/pezzitotale
```

Blocco 3 / 5 (15')

```
% Qual'è l'operaio più efficiente?
% Per rispondere a questa domanda si crea un vettore efficienza, dove ogni
% elemento valuta l'efficienza del singolo operaio. L'efficienza è definita
% come il rapporto tra pezzi prodotti dall'operaio e dalle ore totali
% impiegate per produrre quei pezzi; occorre pertanto un'operazione di
% divisione elemento per elemento tra i vettori 'pezzi' e 'oresett'
efficienza=pezzi./oresett
% Ora che sono state calcolate tutte le efficienze dei singoli operai,
% occorre individuare l'operaio migliore, ovvero colui che ha efficienza più
% elevata. L'operazione si riduce a ricavare il valore massimo contenuto in
% un vettore (efficienza) e l'indice corrispondente, che individua
% l'operaio. A tal scopo si utilizza la funzione 'max'. help max per
% ulteriori informazioni
% L'operazione max funziona in questo modo:
% [valore_massimo, posizione_valore_massimo] = max(vettore_in_input)
[efficienzamax,chi]=max(efficienza)
```

Succo del discorso, fino ad ora :)

In MATLAB si ragiona con un paradigma diverso dal C:

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

Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base^2
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base²
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Da adesso in avanti, chi può utilizzi MATLAB/Octave

Blocco 4 / 1 (20')

```
% Plottare le seguenti funzioni usando tre passi differenti
% passo 1: 1
% passo 2: 0.01
% passo 3: 0.00001

% PRESTATE ATTENZIONE ALL'USO CORRETTO DI .* E * !

y = exp(-0.4*x)*sin(x), x ∈ [0, 2π]
y = sin(x)+2.5*sin(2*x)-1.2*sin(3*x), x ∈ [0, 10]
y = exp(-0.05*x)*cos(5*x), x ∈ [0, 10]
y = x*cos(5*exp(-0.05*x)), x ∈ [0, 10]
```

Blocco 4 / 1 (20')

```
passo1 = 1
passo2 = 0.01
passo3 = 0.00001
figure
hold on
x1 = [0:passo1:2*pi];
x2 = [0:passo2:2*pi];
x3 = [0:passo3:2*pi];
y1 = exp(-0.4*x1).*sin(x1); % curva 1 occhio al .* ed al *
y2 = exp(-0.4*x2).*sin(x2); % curva 2
y3 = exp(-0.4*x3).*sin(x3); % curva 3
plot (x1, y1)
plot (x2, y2)
plot (x3, y3)
```

Blocco 4 / 1 (20')

```
passo1 = 1
passo2 = 0.01
passo3 = 0.00001
figure
hold on
x1 = [0:passo1:10];
x2 = [0:passo2:10];
x3 = [0:passo3:10];
y1 = sin(x1)+2.5*sin(2*x1)-1.2*sin(3*x1); % curva 1
y2 = sin(x2)+2.5*sin(2*x2)-1.2*sin(3*x2); % curva 2
y3 = sin(x3)+2.5*sin(2*x3)-1.2*sin(3*x3); % curva 3
plot (x1, y1)
plot (x2, y2)
plot (x3, y3)
```

Blocco 4 / 1 (20')

```
passo1 = 1
passo2 = 0.01
passo3 = 0.00001
figure
hold on
x1 = [0:passo1:10];
x2 = [0:passo2:10];
x3 = [0:passo3:10];
y1 = exp(-0.05*x1).*cos(5*x1); % curva 1
y2 = exp(-0.05*x2).*cos(5*x2); % curva 2
y3 = exp(-0.05*x3).*cos(5*x3); % curva 3
plot (x1, y1)
plot (x2, y2)
plot (x3, y3)
```

Blocco 4 / 1 (20')

```
passo1 = 1
passo2 = 0.01
passo3 = 0.00001
figure
hold on
x1 = [0:passo1:10];
x2 = [0:passo2:10];
x3 = [0:passo3:10];
y1 = x1.*cos(5*exp(-0.05*x1)); % curva 1
y2 = x2.*cos(5*exp(-0.05*x2)); % curva 2
y3 = x3.*cos(5*exp(-0.05*x3)); % curva 3
plot (x1, y1)
plot (x2, y2)
plot (x3, y3)
```


Blocco 4 / 1 (20')

```
% Plottare le seguenti funzioni usando tre passi differenti
% passo 1: 1
% passo 2: 0.01
% passo 3: 0.00001
```

```
y = exp(-0.4*x)*sin(x), x ∈ [0, 2π]
y = sin(x)+2.5*sin(2*x)-1.2*sin(3*x), x ∈ [0, 10]
y = exp(-0.05*x)*cos(5*x), x ∈ [0, 10]
y = x*cos(5*exp(-0.05*x)), x ∈ [0, 10]
```

Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base²
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Blocco 5 / 1 (20')

Scrivere il codice che genera questa immagine

Il dominio (i.e.: le coppie) X,Y viene generato con la funzione:

meshgrid

La superficie ($Z=f(X,Y)$) é restituita dalla funzione:

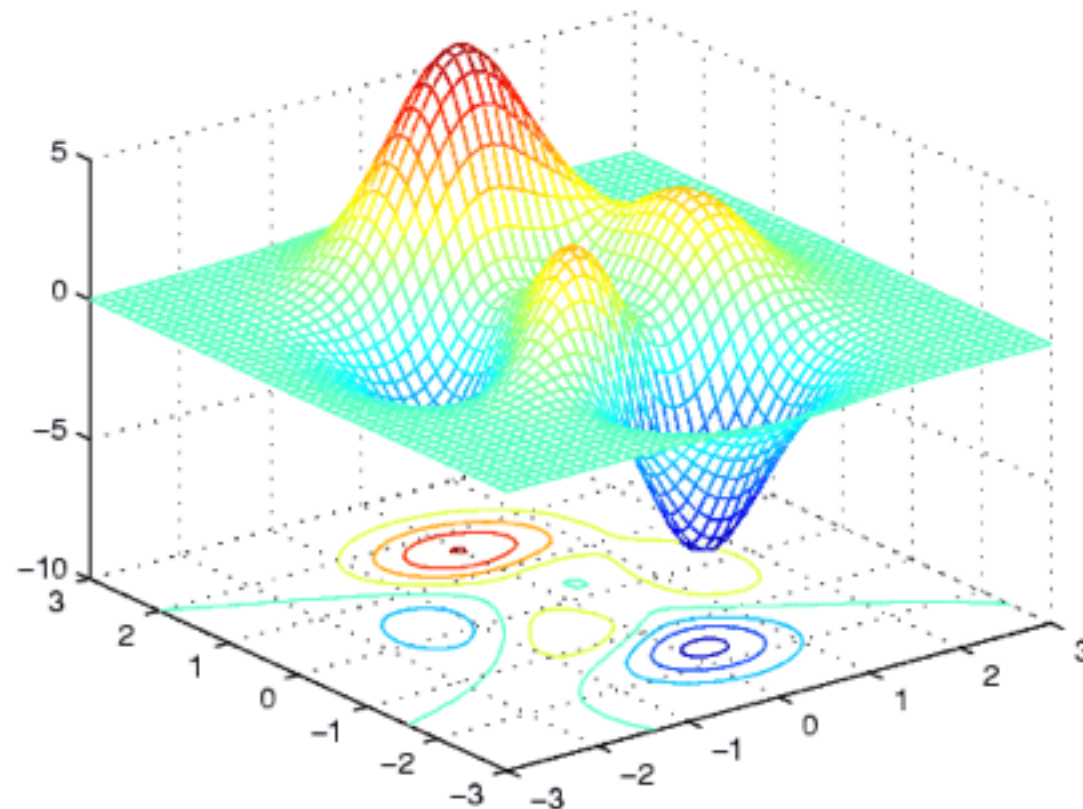
peaks

Il plot 3D della superficie e dei contorni é ottenuta con la funzione:

meshc

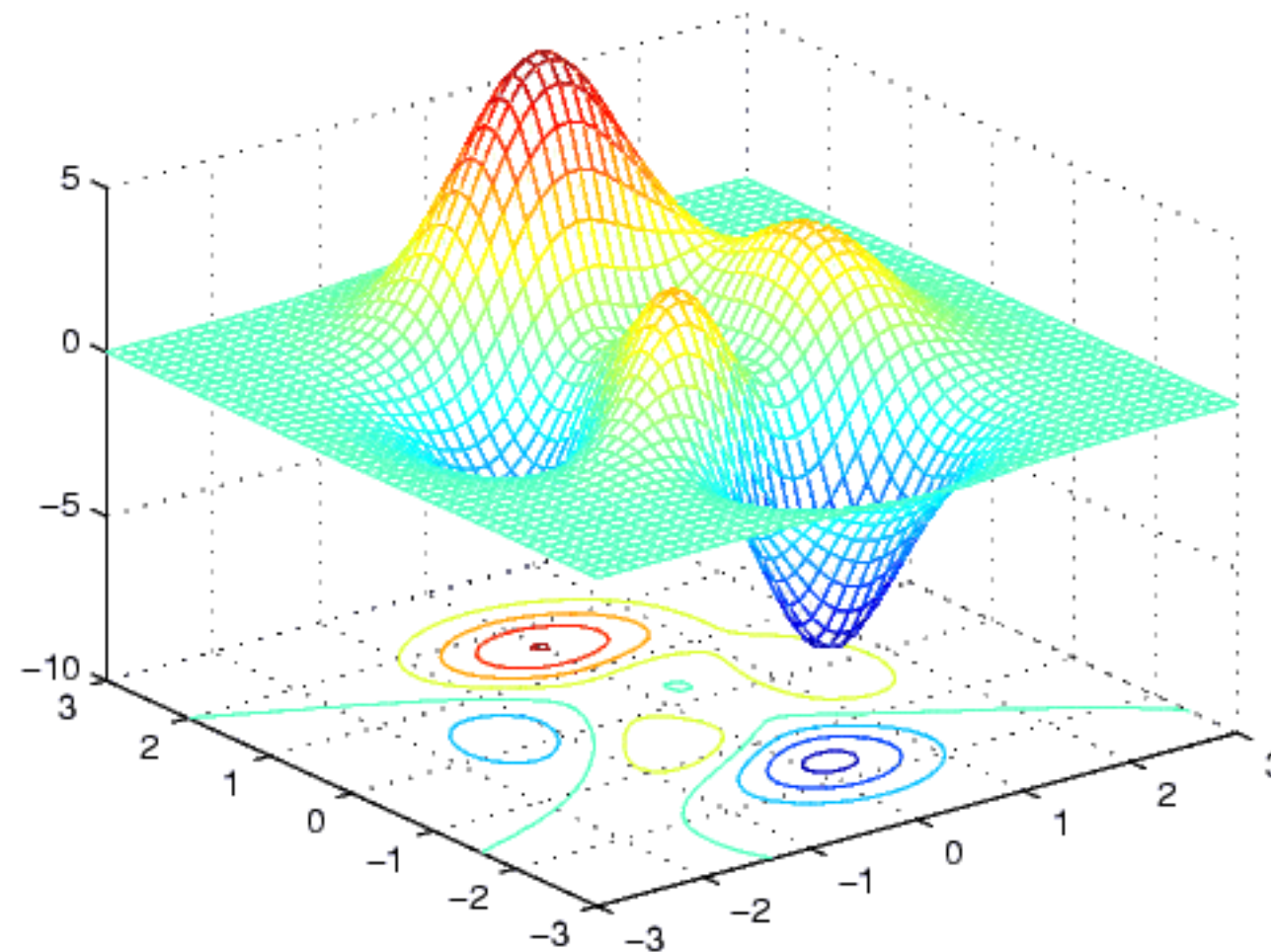
Usate la documentazione MATLAB per ottenere info su come invocare correttamente le funzioni

doc, help



Blocco 5 / 1 (20')

```
[X,Y] = meshgrid(-3:.125:3);  
Z = peaks(X,Y);  
meshc(X,Y,Z);  
axis([-3 3 -3 3 -10 5])
```



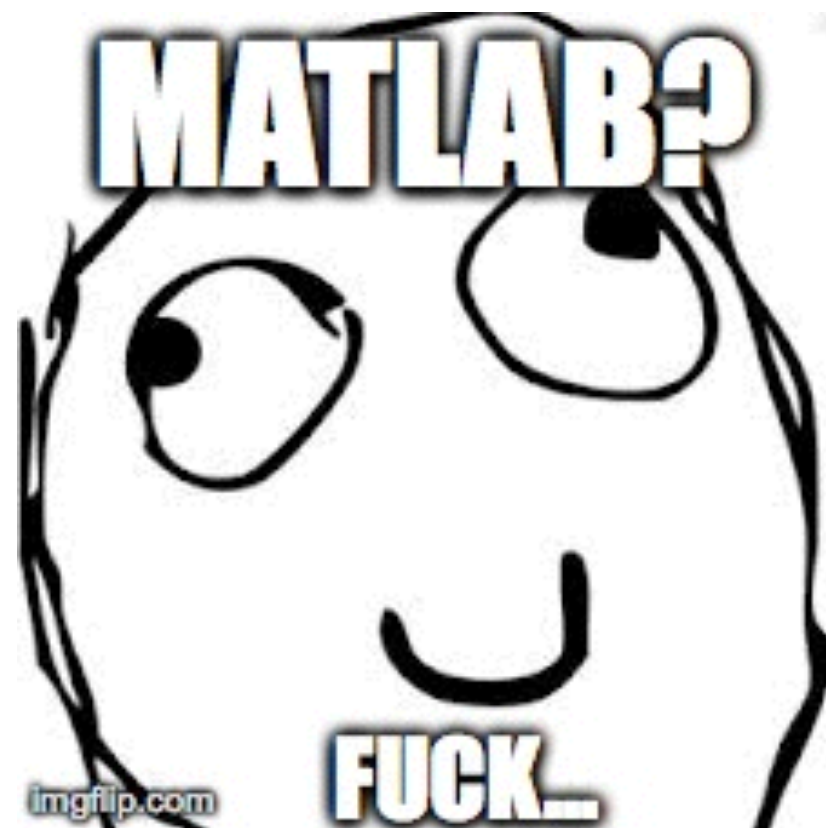
Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base^2
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base^2
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Which one so far? Question time! (10')



Quinta esercitazione: agenda

- (60') Blocco 3: MATLAB base²
- (5') Pausa
- (20') Blocco 4: un po' di plotting 2D
- (20') Blocco 5: un po' di plotting 3D
- (5') Pausa
- (10') Question time
- (50') Blocco 6: MATLAB, strutture dati

Blocco 6 / 1 (15')

```
% Parte 1: creare un array arrayProfessori, un array di struct
% Parte 2: ogni elemento dell'array ha i seguenti campi:
%           1) matricola
%           2) nome
%           3) secondoNome
%           4) cognome
%           4) corsi (una struct; non dichiariamo subito i campi)
```

Blocco 6 / 2 (15')

Commento al codice su MATLAB

Blocco 6 / 2 (15')

- % Parte 3: assegniamo due professori in posizione 1 e 2
- % Parte 4: aggiungere dinamicamente dei campi alla struttura
- % corso, scegliete i campi che preferite
- % Parte 5: analisi dell'occupazione della memoria in tutti gli
- % step intermedi, prevedere e giustificare loro
- % occupazione

Blocco 6 / 2 (15')

Commento al codice su MATLAB