Elaborazione di Segnali e Immagini (ESI)
LABORATORIO

Lezione 1

Manuele Bicego

Corso di Laurea in Informatica

Dipartimento di Informatica - Università di Verona

Informazioni preliminari sul corso di lab

Il docente

Manuele Bicego
Dipartimento di informatica

Ufficio: Ca' Vignal 2 – Primo Piano – Stanza 1.48b
Telefono: 045 8027072
e-mail: manuele.bicego@univr.it

Ricevimento:

- Martedì: 10.30 – 12.30 (in alternativa inviare una email per concordare un appuntamento)

Tutor

Giulia Biolo

Dipartimento di Informatica

Contatti: giulia.biolo@univr.it

Il corso di LAB

Crediti: 2 CFU

Orario: Mercoledì: 8.30 – 11.30 (Lab Delta)

Come funziona il lab

- Il docente introduce l’argomento e descrive gli esercizi
- Gli studenti lavorano sugli esercizi, interagendo con docente e tutor
- Due tipologie di esercizi
  - **Esercizi Principali**: sono molto importanti da portare a termine per capire il tema della lezione
  - **Esercizi Extra**: spiegano altri aspetti interessanti, da fare in aula, se si riesce, e da studiare a casa

Come funziona il lab

- Le soluzioni degli esercizi “principali” verranno presentate all’inizio della lezione successiva

- Tutte le soluzioni vengono comunque messe on line (sia dei “principali” che degli “extra”)

Nota preliminare

- I pc del lab hanno Matlab installato
- In alternativa, si può scaricare e installare Matlab sul proprio pc con la licenza Campus.
- Info su portale **myunivr**:
- → Come fare per → Informatica, wi-fi e applicazioni → Software disponibile → Contratti campus software → Contratto campus software per calcolo matematico MATLAB

Precisazione sulla parte di Lab dell'esame

Dalla lezione introduttiva di teoria:

Modalità d'esame Informatica

- Compito scritto con
  - Parte teorica - domande sui contenuti teorici visti a lezione, punteggio massimo 16/30 punti. Le domande richiederanno definizioni e dimostrazioni, e riprenderanno quanto è stato visto in classe.
  - Parte pratica - consiste di una esercitazione in MATLAB che prevede alcuni esercizi per testare la conoscenza della sintassi di MATLAB e gli operatori fondamentali per l'elaborazione dei segnali e immagini. Punteggio massimo 16/30 punti.
- Il voto finale sarà la somma dei punteggi.

Esercizi da fare sul foglio, sono parte dello scritto

Precisazione sulla parte di Lab dell'esame

- Esercizi di "comprensione del codice".
- Un esempio di esercizio dell'esame scritto:

- Il seguente codice, che implementa il calcolo della media, contiene un errore di natura semantica
- Si indichi dove si trova l'errore e come potrebbe essere corretto (si prega di motivare la risposta)

```
% x è un vettore di numeri
m = 0;
N = length(x);
for i = 1:N
   m = m+x(i);
end
m = m/(N-1);
```

Lezione 1: Matlab

Introduzione a Matlab

- Matlab ha un'interfaccia grafica interattiva e una linea di comando (prompt), sulla quale si possono scrivere dei comandi

```
>> 2+2

ans =

     4
```

Introduzione a Matlab

- Le istruzioni sono frequentemente nella forma:

```
variabile = espressione
```

- Matlab stampa il risultato dell'operazione, a meno che il comando non sia seguito da un punto e virgola.

```
>> a = 2+2     % stampa a video il risutato
>> a = 2+2;    % non stampa
```

- NOTA: “%” serve per inserire un commento

Workspace

Per avere informazioni sulle variabili che sono state inizializzate:

- Opzione 1: tramite interfaccia grafica, controllando la finestra / scheda “workspace”

Workspace

Opzione 2: con i comandi who-whos

```
>> who          % visualizza tutte le variabili
                % definite dall'utente

>> whos         % visualizza tutte le variabili
                % con indicata la loro
                % dimensione
```

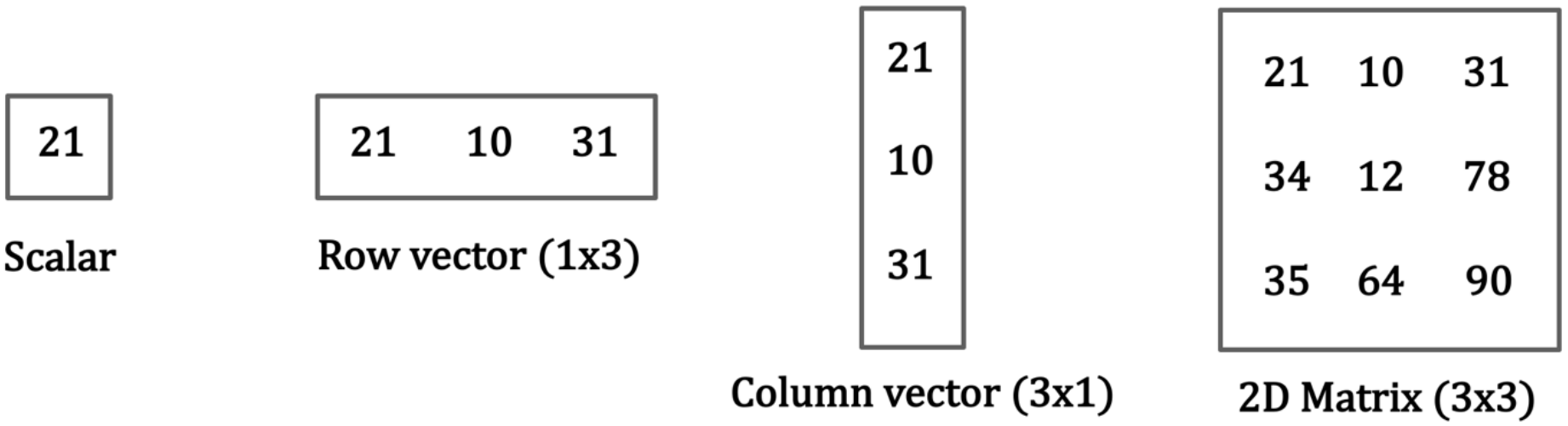
Per cancellare variabili dal workspace

```
>> clear A   % cancella la variabile A

>> clear all % cancella tutte le variabili
```

Definizione di matrici e vettori

- MATLAB lavora sempre con matrici
  - Scalari: matrici 1x1
  - Vettori riga: matrice 1xN
  - Vettori colonna: matrice Nx1



Definizione di matrici e vettori

Definizione matrici:

- le colonne si separano con virgole o spazi
- le righe si separano con punto e virgola
- la matrice si delimita con le parentesi quadre

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

A =

     1     2     3
     4     5     6
     7     8     9
```

Definizione di matrici e vettori

- I vettori si definiscono nello stesso modo

```
>> vettore_riga = [3 5 6]

vettore_riga =

     3     5     6
```

```
>> vettore_colonna = [1;2;56]

vettore_colonna =

     1
     2
    56
```

Definizione di matrici e vettori

- Per verificare la dimensione di una matrice:
  - **`size(matrice)`** → restituisce le dimensioni della matrice
  - **`length(matrice)`** → restituisce la lunghezza del vettore, nel caso di matrici corrisponde alla dimensione massima (ovvero MAX(SIZE(X)))
  - **`numel(matrice)`** → restituisce il # totale di elementi nella matrice

Sottomatrici

- Si può accedere a singoli elementi o sottoparti di una matrice usando gli indici **`(riga,colonna)`**

$$A = \begin{pmatrix} (1,1) & (1,2) & (1,3) \\ (2,1) & (2,2) & (2,3) \\ (3,1) & (3,2) & (3,3) \end{pmatrix}$$

2D Matrix, 3x3 size

```
>> A(2,3)          Estrae il valore corrispondente alla
                   riga = 2, colonna = 3
>> A(:,1)          Estrae tutti i valori nella colonna = 1
>> A(2,:)          Estrae tutti i valori nella riga = 2
>> A(2:3,1:2)      Estrae una sotto-matrice
```

NOTA: i ":" indicano che si accede all'intera riga o all'intera colonna

Sottomatrici

- E' possibile modificare i valori di una matrice nel seguente modo (es. A matrice 3x3):

```
>> A(1,1) = 2  % modifico l'elemento 1,1
>> A(:,3) = [2; 3; 2] % modifico la colonna 3
>> A(1:2,3) = [0; 0] % sotto-colonna 3
>> A(1:2, 2:3) = [1 1; 1 0] % sotto-matrice
```

Occorre prestare attenzione alle dimensioni!

Vettori particolari

- Vettori di punti equispaziati con dimensione arbitrariamente grande

```
>> x = [0:0.2:1] % x = [inizio:passo:fine]

x =

  Columns 1 through 4

         0    0.2000    0.4000    0.6000

  Columns 5 through 6

    0.8000    1.0000
```

Vettori particolari

- **`Linspace:`** per un vettore di numeri linearmente equispaziati

```
>> y = linspace(0,1,5) % linspace(inizio, fine, lunghezza vettore)

y =

         0    0.2500    0.5000    0.7500    1.0000
```

Matrici particolari

```
>> A = eye(n)          % matrice identità nxn

>> A = zeros(n,m)      % matrice di soli zeri nxm

>> A = ones(n,m)       % matrice di soli uni nxm

>> A = rand(n,m)       % matrice aleatoria (valori
                       % compresi tra 0 e 1) nxm
```

Operazioni tra matrici

- Importante: le operazioni fondamentali sono definite fra matrici, che devono avere la dimensione "corretta"

```
>> C = A + B
>> C = A * B
>> C = A / B
>> C = A^3
```

Somma fra matrici: $C_{i,j} = A_{i,j} + B_{i,j}$

Prodotto fra matrici: $C_{i,j} = \sum_k A_{i,k} B_{k,j}$

Divisione fra matrici: $C = AB^{-1}$

Elevamento a potenza: $C = A * A * A$

Operazioni tra matrici

Matlab stampa un messaggio di errore ogni volta che le dimensioni delle matrici non sono corrette rispetto all'operazione che si vuole eseguire.

```
>> [1 2 3] + [10 11 12 13]
Matrix dimensions must agree.
```

```
>> [1 2 3; 4 5 6] * [1; 2]
Error using *
Incorrect dimensions for matrix multiplication. Check
that the number of columns in the first matrix matches
the number of rows in the second matrix. To perform
elementwise multiplication, use '.*'.
```

Altre operazioni importanti

```
>> B = inv(A)     Inversa di una matrice (quadrata)
>> B = det(A)     Determinante di una matrice (quadrata)
>> B = diag(A)    Diagonale di una matrice
>> B = trace(A)   Somma gli elementi sulla diagonale (quadrata)
>> B = eig(A)     Autovalori di una matrice
>> B = rank(A)    Rango di una matrice
>> B = sum(A)     Somma degli elementi colonna per colonna
>> B = prod(A)    Prodotto degli elementi colonna per colonna
>> B = min(A)     Minimo degli elementi colonna per colonna
>> B = max(A)     Massimo degli elementi colonna per colonna
>> B = mean(A)    Media degli elementi colonna per colonna
```

Help in linea

Matlab offre un help in linea molto completo.

```
>> help comando
```

Per avere una spiegazione più dettagliata:

```
>> doc comando
```

Script e funzioni

- Uno script è un insieme di comandi Matlab.
- Gli script sono semplici file di testo con estensione “.m”
- Per eseguire uno script è necessario che la cartella dove si trova Matlab sia la stessa dove è contenuto il file .m
- Per eseguire uno script è possibile:
  - Digitare il nome del file (senza “.m”) nella linea di comando
  - Premere F5 dall'editor Matlab
  - Cliccare su “Run” nella barra in alto

Funzioni

- Una funzione Matlab è una lista di comandi che necessita di variabili di input per essere eseguita e restituisce variabili di output.
- Una funzione è contenuta in un file ".m" che ha lo stesso nome della funzione stessa
- Il file che contiene la funzione DEVE iniziare con:

```
function [output] = nome_function (input)
```

- Tutte le variabili definite internamente sono locali (non verranno mostrate nel workspace)

Funzioni: alcuni suggerimenti

- Le righe dopo la prima, se commentate con “%”, documentano la funzione e sono mostrate quando si digita **`help nome_function`** dal prompt di MATLAB

```
function [output] = nome_function (input)
% [output] = nome_function (input)
%
% funzione di prova:
% input -> ingresso della funzione
% output -> output della funzione
%

output = ...
```

Funzioni: alcuni suggerimenti

```
>> help nome_function
  [output] = nome_function (input)

  funzione di prova:
  input -> ingresso della funzione
  output -> output della funzione
```

- per poter eseguire la funzione, questa deve essere richiamata nel prompt di Matlab (o all'interno di uno script o di un'altra funzione)

```
>> input = [7 2 4];
>> output = nome_function(input)
```

Istruzioni di controllo

- **For**: per ripetere un insieme di istruzioni per un numero predeterminato di iterazioni. Deve terminare con end
  - La sintassi generale:

```
for index = values
    statements
end
```

Istruzioni di controllo

- Esempi:

```
for i = [7 10 12 3]
    disp(i); % visualizza
end
```

```
 7

10

12

 3
```

```
 % da 1 a 5
for i = 1:5
    disp(i); % visualizza i
end
```

```
1

2

3

4

5
```

```
 % da 0 a 12 con passo 3
for i = 0:3:12
    disp(i); % visualizza i
end
```

```
 0

 3

 6

 9

12
```

Istruzioni di controllo

- Esempi:

```
% da 10 a 0 con passo -2
for i = 10:-2:0
    disp(i); % visualizza i
end
```

```
10
8
6
4
2
0
```

```
% per valori generici
for i = [1 5 8 17]
   disp(i)
end
```

```
1
5
8
17
```

Istruzioni di controllo

- **`while`**: per ripetere un insieme di istruzioni fino a quando una condizione rimane vera. Deve terminare con end
- La sintassi generale:

```
while expression
    statements
end
```

Esempio

```
i = 1;
while i<=10
    disp(i)
    i = i+3;
end
```

```
 1

 4

 7

10
```

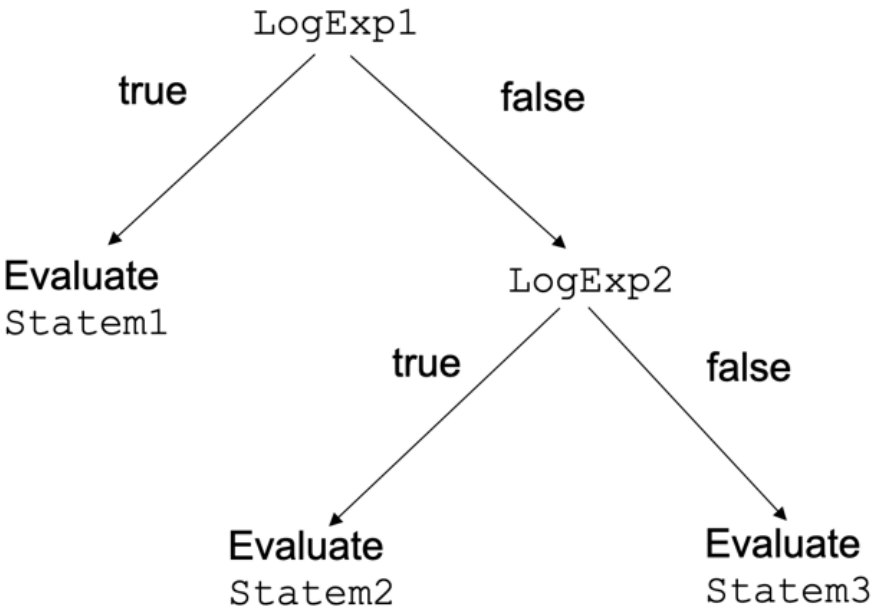
Istruzioni di controllo

- **`if`**: istruzione condizionale, ovvero esegue un'istruzione soltanto se una certa espressione logica è vera. Deve terminare con end.

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

Istruzioni di controllo

```
if logical_expression1
     statement1
elseif logical_expression2
     statement2
else
     statement3
end
```




```
% calcolo la radice di r
% solo se r è positivo
if (r > 0)
     radice = sqrt(r);
else
     radice = NaN;
end
```

Operatori relazionali

MATLAB Operator	Operation	MATLAB Expression
<	Less than	`x<y`
>	Greater than	`x>y`
<=	Less than or equal	`x<=y`
>=	Greater than or equal	`x>=y`
==	Equal	`x==y`
~=	Not equal	`x~=y`

Operatori logici

MATLAB Operator	Operation	MATLAB Expression
~	Not	`~x`
&	And	`x&y`
|	Or	`x|y`
xor	exclusive-Or	`xor(x,y)`

x	y	~x	x&y	x|y	xor(x,y)
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Condizioni su vettori e matrici

- E' possibile effettuare operazioni logiche su vettori e/o matrici

```
>> x = [2 -1 8 0];
>> x > 0

ans =

  1×4 logical array

   1   0   1   0
```

Condizioni su vettori e matrici

- Importante: è possibile usare il risultato di un operazione logica su un vettore per accedere agli elementi il cui risultato è vero

```
>> x = [2 -1 8 0];
>> idx = x > 0;
>> x(idx) = 100

x =

   100    -1   100     0
```

Condizioni su vettori e matrici

- Alternativa: comando find (per ottenere gli indici)

```
>> x = [2 -1; 8 0];
>> tmp = x > 0;
>> [riga, colonna] = find(tmp)

riga =

     1
     2


colonna =

     1
     1
```

Ulteriori fonti

- Canale ufficiale della Mathworks:
  https://it.mathworks.com/help/

- Help in linea di Matlab:
  >> doc

Esercizi principali

Esercizio 1

- Definire i seguenti tre vettori:
  - A vettore riga che contiene i numeri pari da 2 fino a 20
  - B vettore riga con tutti i numeri da -22 a -13
  - C vettore riga con 10 valori uguali a 0.
- A partire da questi, effettuare le seguenti operazioni
  - Creare una matrice MatX dove le righe sono costituite da A, B e C (in questo ordine)
  - Verificare e salvare le dimensioni di MatX e il numero di elementi

Esercizio 1

- Estrarre la sotto-matrice che contiene le prime due righe e le prime cinque colonne
- Sostituire la seconda colonna di MatX con il valore 31
- Creare una matrice MatY di numeri reali distribuiti in modo random (randn), con 4 righe e 10 colonne
- Creare una matrice MatZ data dalla concatenazione di MatX, MatY e di nuovo MatX
- Verificare le dimensioni di MatZ ed estrarre la diagonale.

Esercizio 2

- Generare un numero casuale con il comando randn (distribuzione normale standard)
- Assegnare alla variabile y il valore 1 se tale numero e' compreso tra -1 e 1 (media +- deviazione standard), 0 altrimenti.
- Se ripeto il procedimento 10000 volte, quante volte il numero casuale cade nell'intervallo [-1 1]?
- EXTRA: Provare a risolvere l'esercizio anche senza usare cicli (suggerimento: consultate l'help della funzione randn)

Esercizio 3

- Creare una function chiamata **`Mymean`** che dato un vettore o una matrice in ingresso restituisca il valore medio.
  - Ricorda: la funzione **`Mymean`** deve essere definita in un file che si chiama **`Mymean.m`** e deve iniziare con la seguente riga:

```
function [output] = Mymean (input)
```

  - Dove **`input`** e **`output`** sono rispettivamente l'ingresso e l'uscita della funzione

Esercizio 3

- In particolare, nel caso di vettori la funzione Mymean restituisce un singolo valore medio, mentre per le matrici un vettore riga contenente il valor medio di ogni colonna.
- Controllare che la funzione dia il risultato atteso (confronto con il risultato della funzione **`mean`** di Matlab) con in ingresso un vettore riga, un vettore colonna e una matrice
- Esempio

```
vec = [1:2:30];
vec_media = MYmean(vec);
media_Matlab = mean(vec);
confronto = [vec_media; media_Matlab]
```

Esercizi extra

Esercizio 4

- Inizializzare due vettori v1 e v2 con i valori [0 0] e [1 1] rispettivamente.
- Assumendo che i due numeri contenuti in un vettore siano coordinate (x,y) in un piano cartesiano, calcolare la distanza euclidea tra v1 e v2 (Comandi utili: sum, sqrt)
- Ripetere l'esercizio inizializzando v1 con [2; 0] e v2 con [0; 2].
- Domanda. E' importante che i vettori siano definiti in riga o in colonna?

Esercizio 4

- Formula della distanza euclidea fra due vettori a e b di lunghezza n:

$$d = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

```
Soluzione 1:  1.4142
Soluzione 2:  2.8284
```

Esercizio 5

- Scrivere una funzione, **`checksym`**, che dia 1 se la matrice inserita è simmetrica, 0 altrimenti.
- Nota: la simmetria è definita per matrici quadrate:
  - Controllare che l'ingresso sia effettivamente una matrice (che non sia un vettore o una matrice n-dimensionale)
  - Controllare che l'ingresso sia una matrice quadrata
- Suggerimento: una matrice simmetrica è uguale alla sua trasposta

Esercizio 5

- Nota: in matlab la trasposta si ottiene con l'apice

```
>> A = [1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> A'

ans =

     1     4
     2     5
     3     6
```

Esercizio 5

```
>> matrice_A = [ 1 2; 2 1]

matrice_A =

     1     2
     2     1

>> res = checksym(matrice_A)

res =

     1
```

```
>> matrice_A = [ 1 2; 1 2]

matrice_A =

     1     2
     1     2

>> res = checksym(matrice_A)

res =

     0
```

```
>> matrice_A = [1 2 3; 4 5 6]

matrice_A =

     1     2     3
     4     5     6

>> res = checksym(matrice_A)
Matrix A is not squared, symmetry not defined
res =

    -1
```