LAB 2: Matlab – concetti avanzati

DEBUG CON MATLAB

MATLAB mette a disposizione un ottimo debugger, altro motivo di pregio del sistema. La modalità di debug si attiva dopo aver salvato il file .m e inserendo un breakpoint nell'editor a sinistra del codice (bullet rossa). Mandando in esecuzione il codice l'esecuzione si fermerà al primo break point, entrando in modalità di debug prima di eseguire quella riga. Da lì, si può proseguire una riga alla volta, oppure fino al prossimo break point, oppure fino alla fine del codice.

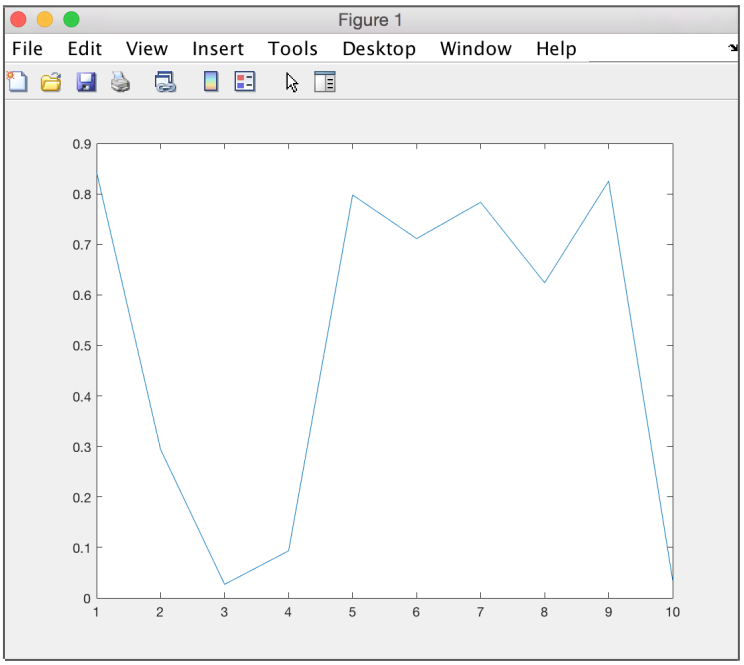
Altre funzionalità utilissime per il debug sono:
`>> dbstop if error;` entra in modalità di debug nel momento in cui riscontra un errore nel codice, in corrispondenza della riga che ha prodotto l'errore, prima di eseguirla. In questo caso si ha la possibilità di ispezionare il workspace alla ricerca di errori;
`>> dbstop if warning;` stessa cosa di sopra, con la differenza che è uno warning ad attivare la modalità di debug
`>> dbquit;` esce dalla modalità di debug
`>> dbclear all;` a volte la modalità di attivazione del debugger con errore diventa stressante! Specialmente quando l'errore è dovuto a dimensionalità sbagliate nelle variabili (sviste più che errori). In questo caso, `dbclear all` annulla le opzioni di attivazione automatica del debugger.

Altre macro per rendere il debugging più veloce ed efficace sono:

- CTRL+r commenta la riga corrente o la porzione di codice selezionata
- CTRL+t leva il primo commento sulla sinistra (se presente) o della porzione di codice selezionata
- CTRL+i indenta la riga di codice o la porzione di codice selezionata guardando l'intero scope dello script o funzione in cui ci si trova

ISTRUZIONI PER LA VISUALIZZAZIONE GRAFICA - SEGNALI

MATLAB ha diversi strumenti per la gestione delle parti grafiche e in particolare è molto adatto per produrre figure varie, inclusi grafici 2D e 3D. Può esportare il risultato (.fig) in diversi formati, come EPS, PDF e JPG. Le figure possono essere inizializzate con il comando `figure(number)`.
Il *line plot* è il tipo di grafico più frequentemente utilizzato per i segnali, richiede due vettori della stessa dimensione, uno per specificare le coordinate orizzontali x di ogni punto, l'altro per definire il valore corrispondente nell'asse delle ordinate y. Quindi un punto generico *i* corrisponde alla coppia *(xi,yi)*. Il comando più semplice è `plot`:

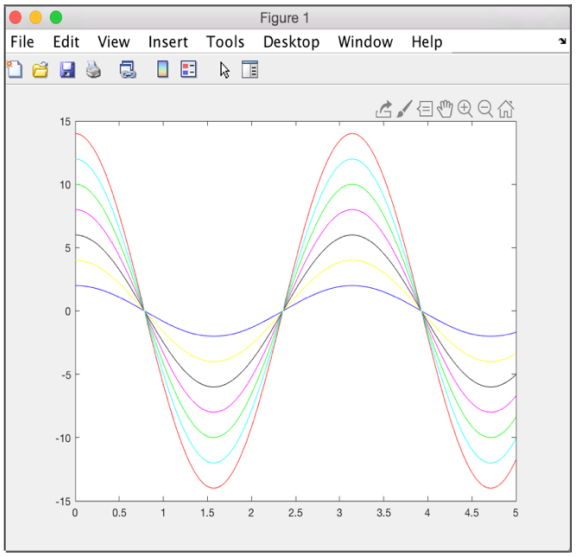



```
>> A = [1:10];
>> B = rand(1,10);
>> plot(A,B)
```

MATLAB di default usa la linea continua blu, senza marker, e senza labels agli assi. Per migliorare la visualizzazione di un grafico creato con plot, ci sono vari elementi che possono essere aggiunti, ad esempio:

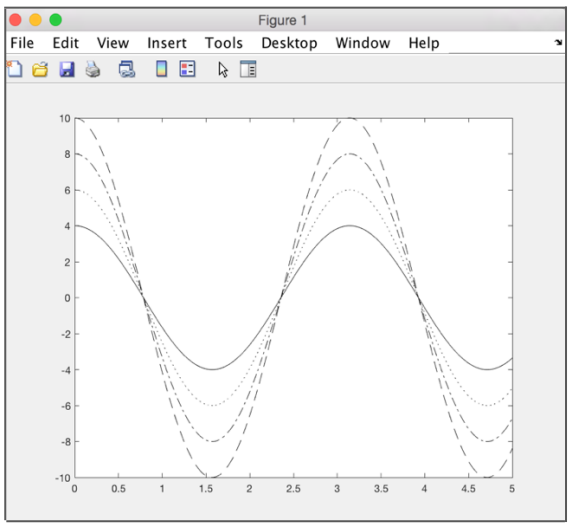
- Linee con uno stile specifico (in termini di dimensioni, marker, colore etc);
- Nomi agli assi e definizione dei loro limiti
- Titolo
- Legenda
- Griglia (eventualmente)

Per il *colore*, MATLAB ha 8 colori predefiniti ma possono essere definiti dall'utente come `plot(x,y,'color',[R G B])` dove i valori devono compresi tra 0 e 1.



b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

Per le *linee*, si possono aggiungere marker per ogni punto e cambiare la tipologia da continua a tratteggiata, ad esempio con `plot(x,y,'k--')` si crea una linea nera tratteggiata. Inoltre, anche lo spessore può essere specificato (default value = 1): `plot(x,y,'k-','LineWidth',value)`.



.	point	-	solid
o	circle	:	dotted
x	x-mark	-.	dashdot
+	plus	--	dashed
*	star	(none)	no line
s	square		
d	diamond		
v	triangle (down)		
^	triangle (up)		
<	triangle (left)		
>	triangle (right)		
p	pentagram		
h	hexagram		

Per specificare titolo e nomi degli assi, si possono usare i seguenti comandi:

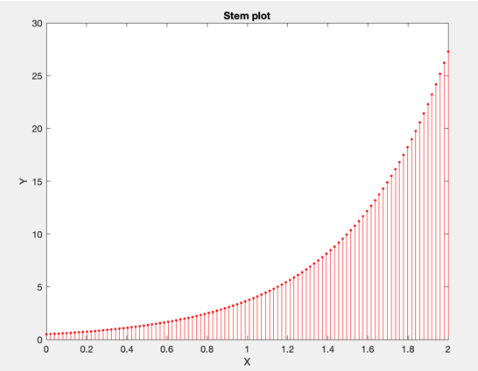
```
>> title('titolo')
>> xlabel('name')
>> ylabel('name')
```

Altri comandi utili sono:

- `axis([xmin xmax ymin ymax])` per settare minimo/massimo per asse
- `xlim([]), ylim([])` per settare minimo/massimo per un asse
- `legend('text')` per visualizzare una legenda a lato
- `grid on/off` per visualizzare (o no) la griglia nel grafico
- `hold on` per visualizzare piu' grafici sovrapposti
- `clf` per pulire il contenuto di una figura
- `stem` per visualizzare la sequenza dei dati Y come steli che si estendono per tutta la lunghezza X
- `subplot` per creare piu' immagini all'interno di una singola figura

Esempio 1:

```
X = linspace(0,2,100)';
Y = [0.5*exp(2*X)];
figure(1)
stem(X,Y,'Color','r','Marker','.')
title ('Stem plot')
xlabel ('X')
ylabel ('Y')
```

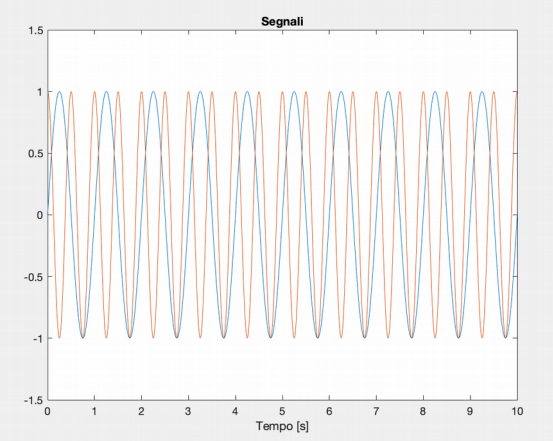


Esempio 2:

```
figure
t = [0:0.01:10]; %crea un vettore da 0 a 10, passo 0.01
ys = sin(2*pi*1*t); %crea un'onda sinusoidale con f = 1Hz
plot(t,ys); % plotta l'onda sinusoidale

yc = cos(2*pi*2*t); % crea un coseno con f = 2 Hz
plot(t,ys,t,yc) % plotta le due onde contemporaneamente
```

```
axis([0 10 -1.5 1.5]) % forza gli estremi degli assi
title('Segnali') % scrive il titolo della figura
xlabel('Tempo [s]') % scrive la label delle ascisse
```



In alcuni casi è utile interagire con i comandi attivabili direttamente dall'interfaccia grafica MATLAB.
Inoltre, le proprietà di un'immagine si possono modificare attraverso i comandi `get` (osserva particolari proprietà delle immagini) e `set` (cambia tali proprietà su valori decisi da utente). Ad esempio, una utile macro è:

```
>>    set(gcf,'Name','NOME    DA    INSERIRE    PER    LA
FIGURA','IntegerHandle','off');
```

In pratica, questa operazione setta alcune proprietà della figura come desideriamo noi, ad esempio permette di dare un nome alla figura e inoltre non attribuisce più alla figura un numero intero, come è solito fare MATLAB.

Breve Esercizio: Provate a rappresentare i due segnali in due plot separati in una stessa figura (suggerimento – comando subplot)

ISTRUZIONI PER LA VISUALIZZAZIONE GRAFICA – SUONI

I suoni sono segnali che si possono visualizzare.
Per **leggere** un file contenente un suono, MATLAB mette a disposizione il comando `[Y, FS] = audioread (FILENAME)` che legge un file audio specificato dalla stringa FILENAME, restituendo i dati campionati in Y e la frequenza di campionamento FS, in Hertz.

Per esempio, si provi (probabilmente in Delta non funziona, fate una prova):

```
>> [y,Fs] = audioread('400SineWave.mp3');
```

Il suono lo si può poi ascoltare attraverso la funzione `sound`:

```
>> sound(y(1:Fs*0.5,:),Fs)
```

Lo si può visualizzare attraverso la funzione `plot`:

```
t = 1:size(y(1:Fs/2,1),1);
t = t./Fs;
figure; plot(t,y(1:Fs/2,1))
xlabel('t [sec]')
ylabel('amplitude')
```

Si può provare anche con un altro suono:

```
[Y1,fs1] = audioread('funky.mp3');
sound(Y1(1:fs1*4,:),fs1)
figure;
set(gcf,'name','Dataset
canzoni','IntegerHandle','off');
plot(Y1(1:fs1*4,1),'-k','LineWidth',2);
title ('Suono Funky.mp3')
xlabel('t [sec]')
ylabel('amplitude')
grid on
```

MATLAB (superiore alla versione 2015) mette a disposizione anche un audio recorder, istanziabile attraverso la funzione `audiorecorder`:

```
>> r = audiorecorder(44100,16,1)
```

che indica di preparare il recorder per catturare un suono a 44100 Hertz, 16 bit, un canale (suono monaurale, ossia non stereo).
Successivamente registro, ascolto e visualizzo il suono catturato:

```
recordblocking(r,5); % registro 5 secondi
play(r)      % ascolto
```

```
doubleArray = getaudiodata(r);  % estraggo il segnale
                                % audio dal recorder
figure
plot(doubleArray);    % visualizzo
title('Audio Signal (double)');
xlabel('t [sec]')
ylabel('amplitude')
grid on
```

ISTRUZIONI PER LA VISUALIZZAZIONE GRAFICA - IMMAGINI

Per **leggere** un file contenente una immagine in scala di grigi o a colori, MATLAB mette a disposizione il comando:

```
>> I = imread (filename,fmt);
>> [I,map] = imread (...) ;
>> [I,map] = imread (filename);
>> [I,map] = imread (URL ,...) ;
```

Viene letta un'immagine di formato 'fmt' e di nome (e percorso) specificato da 'filename', oppure caricata dal web dall'indirizzo specificato da 'URL' e viene restituita una matrice I con le seguenti caratteristiche:

- di dimensioni M × N se a scala di grigio oppure se immagine binaria, contenente solo valori discreti 0 e 1 (dove M × N sono i pixels delle M righe e N colonne);
- di dimensioni M × N × 3 se a colori rappresentati con il modello RGB o HSV;
- di dimensioni M × N × 4 se rappresentati con il modello CMYK (tipico dei file *.tiff).

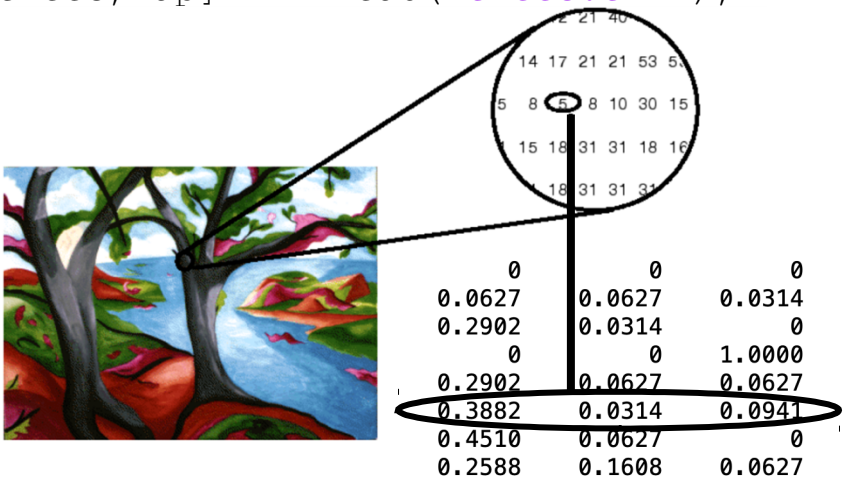
Se il formato `"fmt"` non viene specificato, la funzione `imread` cerca di capire il formato dal file stesso, quindi non è un campo obbligatorio.
Se la variabile `map` è nulla, si parla di *immagine di intensità*, in caso contrario stiamo lavorando con una *immagine indicizzata* (MxN pixels), la cui relativa mappa di colore è salvata in `map`. Una mappa di colore serve per indicare a MATLAB quale colore mostrare a schermo in relazione ad un particolare valore numerico. Esistono mappe colori diverse.

Un' **immagine indicizzata** è composta da una matrice di dati, *X* e una matrice di colori, *map*. `map` è un array m-by-3 di double contenente valori a virgola mobile nell'intervallo [0, 1]; ogni riga specifica i componenti rosso, verde e blu di un singolo colore.
Un'immagine indicizzata utilizza la "mappatura diretta" dei valori dei pixel ai valori della mappa di colori. Il colore di ciascun pixel dell'immagine viene determinato mappando il valore di X al corrispondente colore nella mappa di colori (ovvero serve come indice nella mappa). I valori di X quindi devono essere numeri interi. La mappatura dipende dalla classe della matrice X:

- se X è di tipo *single o double*, i valori in X sono interi nel range [1, p] dove p è la lunghezza della mappa di colori. Il valore 1 punta alla prima riga nella mappa, il valore 2 punta alla seconda riga e così via;
- se X è di tipo *logico, uint8 o uint16*, i valori in X sono interi nel range [0, p-1]. Il valore 0 punta alla prima riga nella mappa, il valore 1 punta alla seconda riga e così via.

Ad esempio, un'immagine indicizzata in MATLAB è:

```
>> [Itrees,map] = imread('trees.tif');
```



Questo è un esempio di immagine indicizzata, in cui vengono visualizzate anche la matrice `Itrees` e la mappa `map`. `Itrees` è di tipo uint8, quindi il valore 5 corrisponde alla sesta riga della mappa di colori. Infatti, sono presenti nella matrice anche valori `Itrees = 0` che corrispondono alla prima riga della mappa (RGB 0,0,0), mentre il valore più grande è 127, corrispondente alla riga #128 della mappa di colori (RGB 1,1,1).

Un'**immagine di intensità** è una matrice di dati, I, i cui valori rappresentano intensità all'interno di un intervallo. Un'immagine di intensità è rappresentata come una matrice, con ogni elemento della matrice corrispondente a un pixel dell'immagine. La matrice può essere di classe *double, uint8 o uint16*. Sebbene le immagini di intensità vengano salvate senza una mappa colori, viene comunque utilizzata una mappa dei colori per visualizzarle. Ad esempio, mappe che vengono implicitamente usate per il plot delle immagini di intensità sono *jet* (dal blu=0 al rosso=255, vedremo), oppure *parula* (dal blu=0 al giallo=255, vedremo).
Un'immagine di intensità può essere di dimensionalità MxN (singolo canale, scala di grigi) oppure MxNxK. Se K=3, la matrice viene trattata come un'immagine a colori. Queste vengono definite anche come "truecolor images", tra cui le RGB sono le più comuni. In questo caso, ogni pixel ha un particolare colore descritto dalla quantità di rosso, verde e blu contenuta, quindi per ogni pixel abbiamo 3 valori corrispondenti.

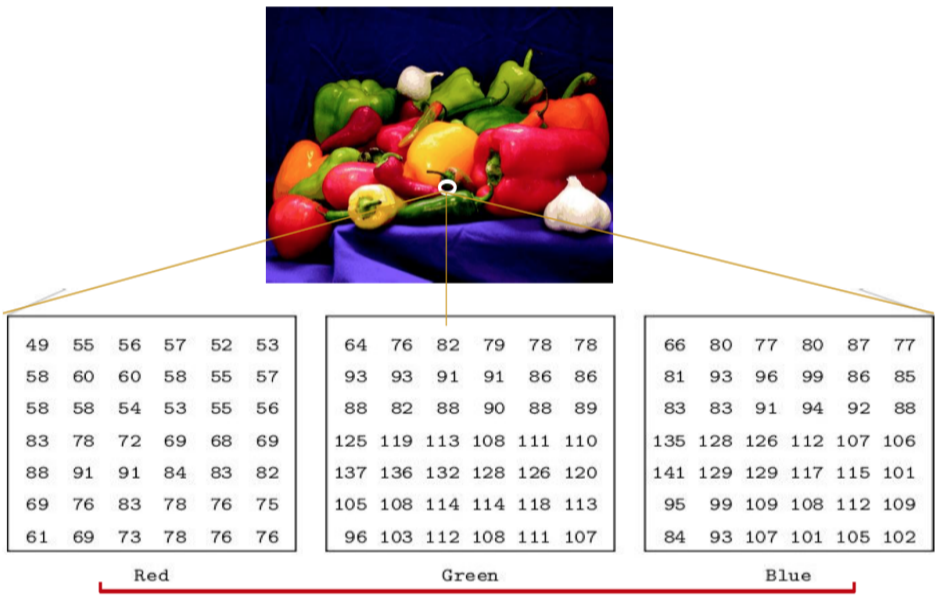
Ad esempio, un'immagine di intensità in MATLAB è:

```
>> [Icam, map] = imread('cameraman.tif');
```

In questo caso, map è vuota ([]), mentre Icam è una matrice 2D, di dimensioni 256x256 di tipo uint8 (quindi valori interi da 0 a 255). Se clicco nel workspace sulla corrispondente variabile posso esplorane il contenuto.

Questa è un'immagine a scala di grigi, mentre un esempio di immagine a colori è dato da:

```
>> [Ipep, map] = imread('peppers.png');
```



dove `Ipep` è una matrice 3D, di dimensioni 384x512x3 di tipo unit8, mentre `map` è ancora una volta una matrice vuota.

È possibile ottenere le informazioni su un file di immagine in una struttura info, tramite il comando Info = imfinfo (filename). Ad esempio:

```
>> imfinfo ('autumn.tif')
```

```
               Filename: '/Applications/MATLAB_R2020a.app/toolbox/images/imdata/autumn.tif'
            FileModDate: '13-Apr-2015 13:23:12'
               FileSize: 214108
                 Format: 'tif'
          FormatVersion: []
                  Width: 345
                 Height: 206
               BitDepth: 24
              ColorType: 'truecolor'
        FormatSignature: [73 73 42 0]
              ByteOrder: 'little-endian'
         NewSubFileType: 0
          BitsPerSample: [8 8 8]
            Compression: 'Uncompressed'
PhotometricInterpretation: 'RGB'
           StripOffsets: [1×30 double]
        SamplesPerPixel: 3
           RowsPerStrip: 7
        StripByteCounts: [1×30 double]
            XResolution: 72
            YResolution: 72
         ResolutionUnit: 'Inch'
               Colormap: []
     PlanarConfiguration: 'Chunky'
              TileWidth: []
             TileLength: []
            TileOffsets: []
         TileByteCounts: []
            Orientation: 1
              FillOrder: 1
      GrayResponseUnit: 0.0100
        MaxSampleValue: [255 255 255]
        MinSampleValue: [0 0 0]
          Thresholding: 1
                Offset: 213642
      ImageDescription: 'Copyright The MathWorks, Inc.'
```

Per **visualizzare** una immagine in scala di grigi o a colori contenuta in una matrice I, MATLAB mette a disposizione una serie di comandi. Il comando principale è `imshow`:

```
>> imshow (I);              % Per scala di grigi o RGB
>> imshow (I,[low high]);   % Per scala di grigi
>> imshow (I,[]);           % Per scala di grigi
>> imshow (I,map);          % Per immagine indicizzata
```

Nel primo caso, `imshow` usa per la visualizzazione un range di valori di default a seconda del tipo di dato. Nel secondo caso, invece, si va a specificare il range di valori per la visualizzazione, utilizzando un vettore con due elementi (low/high). In una immagine in scala di grigi, in questo modo posso visualizzare solo i pixels con valori all'interno dell'intervallo [low high], gli altri valori saranno sostituiti con il colore nero, se il loro valore è minore di low, altrimenti, se maggiore, con il colore

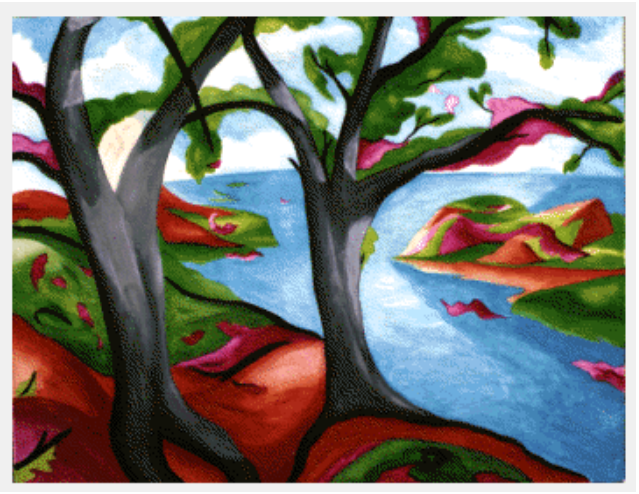
bianco. Se uso [], `imshow` scala la visualizzazione sulla base dei valori dei pixels nell'immagine I.
Posso anche visualizzare una immagine indicizzata con la relativa mappa di colore contenuta nella variabile `map`, oppure, se serve, può restituire l'handle della figura (`handle = imshow (...);`)

Ad esempio, con il seguente comando viene eseguito per la visualizzazione un clipping dei valori di un'immagine di intensità:

```
I = imread ('cameraman.tif');
figure (1)
subplot (1,2,1), imshow (I); colorbar
subplot (1,2,2), imshow (I,[0 80]) ; colorbar
```

Al contrario, per una immagine indicizzata devo fornire come input sia la matrice che la mappa di colori:

```
[Itrees,map] = imread('trees.tif');
figure(2)
imshow(Itrees,map);
```



Un ulteriore modo alternativo per visualizzare una immagine in scala di grigi o a colori contenuta in una matrice A (quindi trattata come un'immagine), è quello di utilizzare il comando `imagesc(matrix)`.
Nel caso di immagine a canale singolo, l'immagine ottenuta è visualizzata utilizzando tutto il range di colori compreso nella mappa di colore (colormap). Ovvero se un'immagine ha valore minimo `m` e massimo `M`, e supponendo una colormap gray, `m` verrà mappato a nero, `M` a bianco, e tutti i valori intermedi saranno interpolati a 255 valori. Questo perché `imagesc` nasce per visualizzare una generica informazione bidimensionale, non limitandosi alle immagini; pertanto, uno dei suoi obiettivi è di fare percepire al meglio l'informazione visuale massimizzando l'utilizzo della mappa cromatica.

Nota: Non usate la funzione `image`, è una versione semplificata di `imagesc`, che però non è utile ai nostri scopi.

Esempi:

```
% Esempio di imagesc con immagine grayscale
I = imread ('cameraman.tif');
figure(1)
imagesc (I); colormap gray


% Esempio di imagesc con immagine RGB
Ipep = imread ('peppers.png');
figure(2)
imagesc (Ipep)
```

Un altro comando utile per la visualizzazione di immagini è `imtool`, che apre una applicazione denominata Image Viewer:

```
>> imtool (I);                % Per scala di grigi o RGB
```

```
>> imtool (I,range);        % Per scala di grigi
>> imtool (I,map);          % Per immagine indicizzata
```

Nota: È importante fare attenzione allo scambio tra le coordinate x ed y necessario perselezionare o visualizzare correttamente la porzione dell'immagine scelta. Ovvero, quando vado a faccio `imtool/imshow/imagesc` e vado a vedere l'immagine, mi da ad esempio X = 350, Y = 258 nell'angolo in basso a destra, quindi nella visualizzazione dell'immagine considera prima le colonne come X e poi le righe come Y. Al contrario, se devo fare l'accesso alla matrice devo prima specificare le righe e poi le colonne quindi `matrice(258,350)`.

Nota: Un comando non consono per le immagini, ma che viene utilizzato spesso ed è efficace, è il comando `surf`:

```
%% Esempio di visualizzazione tramite surf
figure;
surf (I);
shading interp
colormap bone
```

Dove il comando `shading interp` rende più gradevole la visualizzazione 3D, mentre `colormap bone` cambia la mappa di indici di visualizzazione. Surf permette di visualizzare un'immagine come un rilievo geografico, con valli e picchi. Una rappresentazione di questo tipo è efficace per capire l'analisi in frequenza, per vedere l'effetto di operazioni quali estrazioni di edge, segmentazioni etc.

Altri comandi utili sono:
`>> B = imresize(A,scale)` Questo permette di scalare un'immagine (RGB o in scala di grigi), rimpicciolendola oppure ingrandendola a seconda del valore del parametro scale. Se scale è tra 0-1, B<A, al contrario se è maggiore di 1, B>A come dimensioni.
Questa operazione si può ottenere anche manualmente andando a specificare quante righe e colonne vogliamo prendere dell'immagine, ad esempio `A(1:passo:end, 1:passo:end);`
`>> J = imrotate(I,angle)` Questo permette di ruotare un'immagine, in senso antiorario, oppure in senso orario se il valore di angle è negativo. Angle è espresso in gradi;
`>> J = imcrop(I)` Questo visualizza l'immagine e apre un tool iterativo per selezionarne una porzione. Se conosco le righe/colonne che voglio selezionare, posso

in maniera alternativa andare direttamente a salvarle in una nuova matrice, ad esempio `I(righe, colonne);`
`>> I = rgb2gray(RGB)` per convertire una immagine da RGB a scala di grigi
`>> I2 = im2double(I)` per convertire da interi a formato double nel range [0, 1]. Si può applicare a immagini grayscale e RGB.

Per **scrivere** un file contenete una immagine in scala di grigi o a colori, MATLAB mette a disposizione il comando `imwrite`:

```
>> imwrite (I, 'filename', 'fmt');
>> imwrite (I, map, 'filename', 'fmt');
```

Tale comando salva sul disco un file di nome '`filename`' e di formato specificato da `'fmt'` contenente l'immagine rappresentata dalla matrice I. Se specifico `fmt` (non obbligatorio), l'estensione indicata nel nome del file non viene più considerata.
La matrice I può avere dimensioni M x N per immagini in scala di grigi, oppure dimensioni M x N x 3 per immagini a colori rappresentate dal modello RGB o MxNx4 per immagini a colori rappresentate dal modello CMYK.
La seconda riga salva una immagine indicizzata con la relativa mappa di colore, specificata dalla variabile `map`.