UNIVERSITÀ DEGLI STUDI
DI MILANO

Introduction to Matlab

Ruggero Donida Labati

Dipartimento di Informatica
via Bramante 65, 26013 Crema (CR), Italy
ruggero.donida@unimi.it

Suggested lectures

- HELP
- R. J. Braun, " Beginning Matlab Exercises", Department of Mathematical Sciences, University of Delaware, http://www.math.udel.edu/~braun/M349/Matlab_probs2.pdf

Examples of n-dimensional graphs

Load the file `penny.mat` (Matlab libraries). This file describes the surface of a penny.
Try these functions:

- `imshow()`
- `surf()`
- `mesh()`
- `plot3()`
- `pcolor()`
- `contour()`

Other interesting functions:

- `sphere()`
- `streamribbon()`
- `coneplot()`

Example of fitting funcion

Exercise:

- simulation of a sinusoidal signal $x(n)$
- estimation of the sequency $x(m)$ obtained by sampling the signal $x(n)$
- computation of the signal $x(m+k)$, obtained by linear approximation
- accuracy evaluation

First, a sinusoidal signal is simulated.

```
fs = 10000;                                    % sampling frequency Hz
T = 1/fs;                                      % sampling period
tMax = 0.05;                                   % total time
t = [0:T:tMax];                                % time vector
amplitude = 1;                                 % MAX 1
f1 = 50;                                       % signal frequency Hz
omega1 = 2*pi*f1;                              % angular frequency (rad)
phi = 2*pi*0.75;                               % offset
signal1 = cos(omega1*t + phi) * amplitude;     % sinusoidal signal
```

Sampling

```
TS = T*10;
tS = t(1:10:size(t,2));
signalS = signal1(1:10:size(signal1,2));
```

Plot results

```
figure
plot(t, signal1)                % continuous
hold on
plot(tS, signalS, 'xr')         % sampled
xlabel('Time (ms)');
ylabel('x');
title 'Input signal';
legend('Input signal', 'Sampled signal')
```

Linear interpolation

```
% New time axis
tNewMom = tS(1: size(tS,2)-1);
tNew = tNewMom + 0.5 * TS;

% Linear interpolation
signalNew = interp1(tS, signalS, tNew );
```

Plot results

```
figure
plot(t, signal1)                % input signal
hold on
plot(tS, signalS, 'xr')         % sampled signal
plot(tNew, signalNew, 'ok')     % fitted signal
xlabel('Time (ms)');
ylabel('x');
title 'Results';
legend('Input signal', 'Sampled signal', 'Fitted signal')
```

Accuracy evaluation

```
% points of the input signal in the sampling instants
eElements = 5:10:size(t,2)-1;

% error vector
errorV = abs(signal1(eElements) - signalNew);

% mean
meanE = mean(errorV);

% standard deviation
stdE = std(errorV);
```

Print results

```
% print
fprintf('\nRESULTS\n');
fprintf('Mean error = %f\n', meanE);
fprintf('Standard deviation of the error = %f\n', stdE);
```

Visual analysis

```
% show error
figure
bar(tNew, errorV, 'r')
title 'Error'
xlabel('Time (ms)');
ylabel('\Delta_x');
```

Classification of rectangles in a binary image

Treated arguments:

- Segmentation of a binary image;
- Feature extraction related to the area of the segmented objects;
- Classification of the computed features (3 classes: small, medium, big);
- visualization of the obtained results (textual and graphical representations).

The used classes are defined

```
smallLimit = 1500;
mediumLimit = 5000;
```

The image is loaded

```
img = imread('testRectangles.bmp');
% show the binary image
figure
imshow(img)
title('Input image');
```

Segmentation

```
% Searching of the 8-connected regions (white blobs) in a binary image.
% The resulting image represents every segmented object with a different
% intensity value.
L = bwlabel(img);
figure
imshow(L,[])
colormap jet
title('Segmented rectangles')
```

Feature extraction

```
% Computation of the areas
nEl = max(L(:)); % number of rectangles in L
areas = zeros(1,nEl); % vector composed by nEl elements
for i = 1 : nEl
elementIndexes{i} = find(L == i); % Very useful function! help find
areas(i) = size(elementIndexes{i},1);
end
```

Classification

```
% Classification
smallIndexes = find(areas <= smallLimit);
mediumIndexes = find(smallLimit < areas & areas <= mediumLimit);
bigIndexes = find(areas >= mediumLimit);

% Number of elements appertaining to every class
```

```
smallN = size(smallIndexes,2);
mediumN = size(mediumIndexes,2);
bigN = size(bigIndexes,2);
```

Visualization of the obtained results

```
% Textual visualization of the obtained result
fprintf('\nRESULTS\n')
fprintf('Number of small rectangles = %i\n', smallN)
fprintf('Number of medium rectangles = %i\n', mediumN)
fprintf('Number of big rectangles = %i\n', bigN)

% Graphical visualization of the obtained result
imgClasses = zeros(size(img));
for i = 1: smallN
imgClasses(elementIndexes{smallIndexes(i)}) = 1;
end
for i = 1: mediumN
imgClasses(elementIndexes{mediumIndexes(i)}) = 2;
end
for i = 1: bigN
imgClasses(elementIndexes{bigIndexes(i)}) = 3;
end

figure
imshow(imgClasses,[])
colormap jet
title('Classified rectangles');
```

Feature extraction from images

From Image Processing Toolbox.

Steps:

- load a color image;
- image segmentation;
- feature extraction (roundness).

Read the imput image

```
RGB = imread('pillsetc.png');
imshow(RGB);
```

Image segmentation

```
% Thresholding
I = rgb2gray(RGB);
threshold = graythresh(I);
bw = im2bw(I,threshold);
imshow(bw)

% morphological opening operation for removing objects with size < 30 pixels
bw = bwareaopen(bw,30);

% morphological closing operation for improving the boundaries
se = strel('disk',2);
bw = imclose(bw,se);

% morphological fillbw = imfill(bw,'holes');

imshow(bw)
```

Boundary estimation

```
% boundary estimation
[B,L] = bwboundaries(bw,'noholes');

% Plot
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
for k = 1:length(B)
  boundary = B{k};
  plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
```

Feature extaction and visual results

```
stats = regionprops(L,'Area','Centroid');

threshold = 0.94;

% for all the objects
for k = 1:length(B)

  % boundary of the object k
  boundary = B{k};

  % perimeter
  delta_sq = diff(boundary).^2;
  perimeter = sum(sqrt(sum(delta_sq,2)));

  % area
  area = stats(k).Area;

  % roundness index
  metric = 4*pi*area/perimeter^2;

  % reuslts
  metric_string = sprintf('%2.2f',metric);

  % black circles on the objects
  if metric > threshold
    centroid = stats(k).Centroid;
    plot(centroid(1),centroid(2),'ko');
  end

  text(boundary(1,2)-35,boundary(1,1)+13,metric_string,'Color','y',...
       'FontSize',14,'FontWeight','bold');

end

title(['Metrics closer to 1 indicate that ',...
       'the object is approximately round']);
```

Other exercises

http://www.facstaff.bucknell.edu/maneval/help211/exercises.html