# Introductory Matlab Tutorial

Mohamed Aslan

April 1, 2010

## 1 Introduction

### Variables

To create a variable just use it on the left hand side of an equal sign:

```
x = 10;
z = x^0.25;
```

To print the value of a variable, just type its name without the *semi-colon*:

```
z
```

### Vectors

A vector is a matrix with either one row or one column. A vector is defined by placing a sequence of numbers within square braces.

```
% Horizontal vector
h = [3 1 0];
% Vertical vector
v = [4 ; 5 ; 6];
```

### Matrices

Creating a matrix is the same as a vector, separate the rows of a matrix using *semicolons*:

```
A = [1 2 3 ; 2 3 1 ; 1 10 -1];
```

To transpose a matrix:

```
B = A';
```

Matrix multiplication:

```
A = [3 2 1 ; 2 0 1 ; 1 1 1];
A = [1 6 3 ; 4 3 4 ; 1 3 2];
C = A * B;
```

## Conditional If

Execute statements if condition is true.
if expression1
statements1
elseif expression2
statements2
else
statements3
end

```
A = 5;
B = 6;
C = 0;
if A == B
    C = 1;
elseif A < B
    C = 2;
else
    C = 3;
end
```

## For Loop

The for loop executes statements for a number of times.
for index = start:increment:end
statements
end

```
x = 0;
for k = 1:10
    x = x + k;
end
```

## Math Functions

### Trigonometry

- sin ⟶ Sine
- sind ⟶ Sine in Degrees
- asin ⟶ Inverse sine
- asind ⟶ Inverse sine in Degrees
- cos ⟶ Cosine
- cosd ⟶ Cosine in Degrees
- acos ⟶ Inverse cosine
- acosd ⟶ Inverse cosine in Degrees
- tan ⟶ Tangent
- tand ⟶ Tangent in Degrees
- atan ⟶ Inverse tangent
- atand ⟶ Inverse tangent in Degrees
- sec ⟶ Secant
- secd ⟶ Secant in Degrees
- asec ⟶ Inverse secant
- asecd ⟶ Inverse secant in Degrees
- csc ⟶ Cosecant
- cscd ⟶ Cosecant in Degrees
- acsc ⟶ Inverse cosecant
- acscd ⟶ Inverse cosecant in Degrees
- cot ⟶ Cotangent
- cotd ⟶ Cotangent in Degrees
- acot ⟶ Inverse cotangent

- acotd ⟶ Inverse cotangent in Degrees

```
% Find the sine of angle x = 0.9 radian
x = 0.9;
y = sin(x);
```

## Exponential

- $a \wedge n$ ⟶ a to the power n
- sqrt ⟶ Square root
- log10 ⟶ Base 10 logarithm
- exp ⟶ Exponential (Natural)
- log ⟶ Natural logarithm (ln)

```
% Find the square root of x = 4
x = 4;
y = sqrt(x);
```

## Rounding

- round ⟶ Round towards nearest integer
- floor ⟶ Round towards minus
- ceil ⟶ Round towards plus

```
x = 4.5;
y = round(x);
```

## Other Functions

- abs ⟶ Absolute value

## Statistics Functions

### Min

```
X = [5 5 1 2 4 4];
% Find the minimum
C = min(X);
```

### Max

```
X = [5 5 1 2 4 4];
% Find the maximum
C = max(X);
```

### Mean, Average

```
X = [5 5 1 2 4 4];
% Find the mean of a vector
M = mean(X);
% Find the mean of a matrix
A = [1 2 3; 3 3 6; 4 6 8; 4 7 7];
Z = mean(A);
```

### Median

```
X = [5 5 1 2 4 4];
% Find the median
M = median(X);
```

### Mode

```
X = [5 5 1 2 4 4];
% Find the mode
M = mode(X);
```

### Standard Deviation

```
X = [5 5 1 2 4 4];
% Find the standard deviation
S = std(X);
```

## 2 Graphs

### Plotting

```
X = [1 2 3 4 5];
Y = [1 4 9 16 25];
% Plot X against Y for function Y = X^2
plot(X, Y);
% Try plot(X, Y, '*') and plot(X, Y, '-r.')
% Set the graph's title
title('Y=X^2');
% Set the labels for X-axis and Y-axis
xlabel('X');
ylabel('Y');
```

### Bar Chart

```
z = [1 2 3 4 6 4 3 4 5];
bar(z);
xlabel('Index');
ylabel('Value');
```

## 3 Image Processing

### Read Image

```
% Load the image
im = imread(filename);
```

### Display Image

```matlab
% Display the image
imshow(im);
```

## Get Size of Image

```matlab
% Get height and width of the image
[y, x] = size(im);
```

## Complement Image

```matlab
% Complement the image
new_img = imcomplement(img);
```

## RGB to Grayscale

```matlab
% Convert RGB image to grayscale image
im_gray = rgb2gray(im_rgb);
```

## Grayscale to BW

```matlab
% Convert grayscale image to black and white image with threshold of 0.5
im_bw = im2bw(im_gray, 0.5);
% To calculate the threshold automatically and then convert to black and white image
thr = graythresh(im_gray);
im_bw = im2bw(im_gray, thr);
```

## Save Image

```matlab
% Save image to a BMP file
imwrite(im, 'filename.bmp', 'bmp');
% Save image to a JPEG file
imwrite(im, 'filename.jpg', 'jpg');
```

### Crop Image

```
% Creates an interactive Crop Image tool associated with the image
im = imcrop
% Crops the image and specifies the size and position of the crop rectangle
im_crop = imcrop(im, [x y w h]);
```

### Subtract Images

```
% Subtracts one image from another
im = imsubtract(im1, im2);
```

### Entropy of Grayscale Images

```
% Calculates the entropy of grayscale image
j = entropy(im);
```

### Image Histograms

```
% Displays the histogram of image
imhist(im);
```

## 4 Video Processing

### Acquire Video

```
% Select video device
cam = videoinput('winvideo', 3);
% Display the live video
preview(cam);
% Take the snapshot
im = getsnapshot(cam);
```

# 5 Audio Processing

## Acquire Audio

```
    recorder = audiorecorder;
    % Record voice for 5 seconds
    recordblocking(recorder, 5);
    % Play the recording
    play(recorder);
    % Store sound data in array
    snd = getaudiodata(recorder);
    % Plot audio waveform
    plot(snd);
```

## Load Audio (from Wav file)

```
    snd = wavread('song.wav');
    % Plot audio waveform
    plot(snd);
```

# 6 2D-Filters

- average: Averaging filter
- gaussian: Gaussian lowpass filter
- laplacian: Approximates the two-dimensional Laplacian operator
- motion: Approximates the linear motion of a camera
- sobel: Sobel horizontal edge-emphasizing filter
- unsharp: Unsharp contrast enhancement filter

## Average

```
% Choose the filter
f = fspecial('average', [3, 3]);
% Pass through average filter
filtered_obj = imfilter(obj, f);
```

## Gaussian

```
% Choose the filter, with sigma = 0.5
f = fspecial('gaussian', [3, 3], 0.5);
% Pass through gaussian filter
filtered_obj = imfilter(obj, f);
```

## Laplacian

```
% Choose the filter, with alpha = 0.2 (between 0.0 to 1.0)
f = fspecial('laplacian', 0.2);
% Pass through laplacian filter
filtered_obj = imfilter(obj, f);
```

## Motion

```
% Choose the filter, with len = 2 pixels and theta = 0
f = fspecial('motion', 2, 0);
% Pass through motion filter
filtered_obj = imfilter(obj, f);
```

## Sobel

```
% Choose the filter
f = fspecial('sobel');
% Pass through sobel filter
filtered_obj = imfilter(obj, f);
```

## Unsharp

```
% Choose the filter, with alpha = 0.2 (between 0.0 to 1.0)
f = fspecial('unsharp', 0.2);
% Pass through unsharp filter
filtered_obj = imfilter(obj, f);
```

# 7 Fourier Analysis

## Discrete Fourier Transform

```
X = [1 2 3 4 5 6];
% Computes the discrete Fourier transform (DFT) of vector X
Y = fft(X);
```

## Inverse Discrete Fourier Transform

```
X = [1 2 3 4 5 6];
% Computes the discrete Fourier transform (DFT) of vector X
Y = fft(X);
% Computes the inverse discrete Fourier transform (IDFT) of vector Y
X = ifft(Y);
```

## 2-D Discrete Fourier Transform

```
X = [1 2 3 ; 4 5 6];
% Computes the 2-D discrete Fourier transform (DFT) of matrix X
Y = fft2(X);
```

## 2-D Inverse Discrete Fourier Transform

```
X = [1 2 3 4 5 6];
% Computes the 2-D discrete Fourier transform (DFT) of matrix X
Y = fft2(X);
% Computes the inverse 2-D discrete Fourier transform (IDFT) of matrix Y
X = ifft2(Y);
```

## Zero Shift

```
% Shift zero-frequency component to center of spectrum
Z = fftshift(Y);
```

# 8 Wavelet Analysis

## Single-level discrete 1-D wavelet transform

```
% Computes 1-level discrete haar 1-D wavelet transform
[A, D] = dwt(X, 'haar');
% Computes 1-level discrete db1 1-D wavelet transform
[A, D] = dwt(X, 'db1');
% Computes 1-level discrete db2 1-D wavelet transform
[A, D] = dwt(X, 'db2');
% Computes 1-level discrete db4 1-D wavelet transform
[A, D] = dwt(X, 'db4');
% Computes 1-level discrete coif4 1-D wavelet transform
[A, D] = dwt(X, 'coif4');
```

## Single-level inverse discrete 1-D wavelet transform

```
% Computes 1-level discrete haar 1-D wavelet transform
[A, D] = dwt(X, 'haar');
% Computes 1-level inverse discrete haar 1-D wavelet transform
X = idwt(A, D, 'haar');
```

## Single-level discrete 2-D wavelet transform

```
% Computes 1-level discrete haar 2-D wavelet transform
[A, H, V, D] = dwt2(X, 'haar');
```

## Single-level inverse discrete 2-D wavelet transform

```
% Computes 1-level discrete haar 2-D wavelet transform
[A, H, V, D] = dwt2(X, 'haar');
% Computes 1-level inverse discrete haar 2-D wavelet transform
X = idwt2(A, H, V D, 'haar');
```

# 9 Artificial Neural Networks

## Feed-Forward Neural Networks

### Example: XOR gate

```matlab
% Create feed-forward network
% Parameters:
% 1- Max and min of the input values (2 inputs)
% 2- Number of layers 2 layers, 1st (hdidden) has 2 neurons, 2nd (output) has 1 neuron
% 3- Activation functions for each layer, sigmoid function is used
net = newff([0 1; 0 1], [2 1], {'logsig', 'logsig'});

% Input is in the columns of the matrix
% 1 1 0 0
% 1 0 1 0
% The 1st input is 1 1, the 2nd is 1 0, the 3rd is 0 1 and the 4th is 0 0
input = [1 1 0 0 ; 1 0 1 0];

% Target matrix (expected output)
target = [0 1 1 0];

% Train the network
net = train(net, input, target);

% Simulate the network
% Test the network with input as input and see if output is the same as target
output = sim(net, input)
```