1 Introduction

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

\[
\begin{align*}
  x &= 10; \\
  z &= x^{0.25} ;
\end{align*}
\]

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

\[
x
\]

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.

\[
\begin{align*}
  \text{\% Horizontal vector} \\
  h &= \begin{bmatrix} 3 & 1 & 0 \end{bmatrix} ; \\
  \text{\% Vertical vector} \\
  v &= \begin{bmatrix} 4 ; 5 ; 6 \end{bmatrix} ;
\end{align*}
\]

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

\[
A = \begin{bmatrix} 1 & 2 & 3 ; 2 & 3 & 1 ; 10 & -1 \end{bmatrix} ;
\]
To transpose a matrix:

\[
B = A';
\]

Matrix multiplication:

\[
A = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix};
\]
\[
A = \begin{bmatrix} 1 & 6 & 3 \\ 4 & 3 & 4 \\ 1 & 3 & 2 \end{bmatrix};
\]
\[
C = A \ast B;
\]

**Conditional If**

Execute statements if condition is true.

if expression1
\[
\text{statements1}
\]
elseif expression2
\[
\text{statements2}
\]
else
\[
\text{statements3}
\]
end

\[
A = 5;
\]
\[
B = 6;
\]
\[
C = 0;
\]
\[
\text{if } A == B \\
\quad C = 1;
\]
\[
\text{elseif } A < B \\
\quad C = 2;
\]
\[
\text{else} \\
\quad C = 3;
\]
end

**For Loop**

The for loop executes statements for a number of times.

for index = start:increment:end
\[
\text{statements}
\]
end

\[
x = 0;
\]
\[
\text{for } k = 1:10 \\
\quad x = x + k;
\]
\[
\text{end}
\]
Math Functions

Trigonometry

- \( \sin \rightarrow \) Sine
- \( \sin d \rightarrow \) Sine in Degrees
- \( \text{asin} \rightarrow \) Inverse sine
- \( \text{asind} \rightarrow \) Inverse sine in Degrees
- \( \cos \rightarrow \) Cosine
- \( \cos d \rightarrow \) Cosine in Degrees
- \( \text{acos} \rightarrow \) Inverse cosine
- \( \text{acosd} \rightarrow \) Inverse cosine in Degrees
- \( \tan \rightarrow \) Tangent
- \( \tand \rightarrow \) Tangent in Degrees
- \( \text{atan} \rightarrow \) Inverse tangent
- \( \text{atand} \rightarrow \) Inverse tangent in Degrees
- \( \sec \rightarrow \) Secant
- \( \sec d \rightarrow \) Secant in Degrees
- \( \text{asec} \rightarrow \) Inverse secant
- \( \text{asecd} \rightarrow \) Inverse secant in Degrees
- \( \csc \rightarrow \) Cosecant
- \( \csc d \rightarrow \) Cosecant in Degrees
- \( \text{acsc} \rightarrow \) Inverse cosecant
- \( \text{acscd} \rightarrow \) Inverse cosecant in Degrees
- \( \cot \rightarrow \) Cotangent
- \( \cot d \rightarrow \) Cotangent in Degrees
- \( \text{acot} \rightarrow \) Inverse cotangent
• acotd → Inverse cotangent in Degrees

1 % Find the sine of angle x = 0.9 radian
2 x = 0.9;
3 y = sin(x);

Exponential
• $a^n$ → $a$ to the power $n$
• sqrt → Square root
• log10 → Base 10 logarithm
• exp → Exponential (Natural)
• log → Natural logarithm (ln)

1 % Find the square root of $x = 4$
2 x = 4;
3 y = sqrt(x);

Rounding
• round → Round towards nearest integer
• floor → Round towards minus
• ceil → Round towards plus

1 x = 4.5;
2 y = round(x);

Other Functions
• abs → Absolute value
Statistics Functions

Min

```matlab
X = [5 5 1 2 4 4];
C = min(X);
```

Max

```matlab
X = [5 5 1 2 4 4];
C = max(X);
```

Mean, Average

```matlab
X = [5 5 1 2 4 4];
M = mean(X);
A = [1 2 3; 3 3 6; 4 6 8; 4 7 7];
Z = mean(A);
```

Median

```matlab
X = [5 5 1 2 4 4];
M = median(X);
```

Mode

```matlab
X = [5 5 1 2 4 4];
M = mode(X);
```
Standard Deviation

```matlab
X = [5 5 1 2 4 4];
S = std(X);
```

2 Graphs
Plotting

```matlab
X = [1 2 3 4 5];
Y = [1 4 9 16 25];
plot(X, Y);
% Try plot(X, Y, '+') and plot(X, Y, '-*')
% Set the graph’s title
% Y = X^2
title('Y = X^2');
xlabel('X');
ylabel('Y');
```

Bar Chart

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

3 Image Processing
Read Image

```matlab
im = imread(filename);
```

Display Image
Display the image
```matlab
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

```matlab
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)

```matlab
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

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

1 \texttt{fspecial(‘gaussian’, [3, 3], 0.5)};
2 \texttt{imfilter(obj, f)};

Laplacian

1 \texttt{fspecial(‘laplacian’, 0.2)};
2 \texttt{imfilter(obj, f)};

Motion

1 \texttt{fspecial(‘motion’, 2, 0)};
2 \texttt{imfilter(obj, f)};

Sobel

1 \texttt{fspecial(‘sobel’)};
2 \texttt{imfilter(obj, f)};

Unsharp

1 \texttt{fspecial(‘unsharp’, 0.2)};
2 \texttt{imfilter(obj, f)};
7 Fourier Analysis

Discrete Fourier Transform

1 \[ X = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}; \]
2 \% Computes the discrete Fourier transform (DFT) of vector X
3 \[ Y = \text{fft}(X); \]

Inverse Discrete Fourier Transform

1 \[ X = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}; \]
2 \% Computes the discrete Fourier transform (DFT) of vector X
3 \[ Y = \text{fft}(X); \]
4 \% Computes the inverse discrete Fourier transform (IDFT) of vector Y
5 \[ X = \text{ifft}(Y); \]

2-D Discrete Fourier Transform

1 \[ X = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}; \]
2 \% Computes the 2-D discrete Fourier transform (DFT) of matrix X
3 \[ Y = \text{fft2}(X); \]

2-D Inverse Discrete Fourier Transform

1 \[ X = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}; \]
2 \% Computes the 2-D discrete Fourier transform (DFT) of matrix X
3 \[ Y = \text{fft2}(X); \]
4 \% Computes the inverse 2-D discrete Fourier transform (IDFT) of matrix Y
5 \[ X = \text{ifft2}(Y); \]

Zero Shift

1 \% Shift zero-frequency component to center of spectrum
2 \[ Z = \text{fftsift}(Y); \]
8 Wavelet Analysis

Single-level discrete 1-D wavelet transform

```matlab
1 % Computes 1-level discrete haar 1-D wavelet transform
2 [A, D] = dwt(X, 'haar');
3 % Computes 1-level discrete db1 1-D wavelet transform
4 [A, D] = dwt(X, 'db1');
5 % Computes 1-level discrete db2 1-D wavelet transform
6 [A, D] = dwt(X, 'db2');
7 % Computes 1-level discrete db4 1-D wavelet transform
8 [A, D] = dwt(X, 'db4');
9 % Computes 1-level discrete coif4 1-D wavelet transform
10 [A, D] = dwt(X, 'coif4');
```

Single-level inverse discrete 1-D wavelet transform

```matlab
1 % Computes 1-level discrete haar 1-D wavelet transform
2 [A, D] = dwt(X, 'haar');
3 % Computes 1-level inverse discrete haar 1-D wavelet transform
4 X = idwt(A, D, 'haar');
```

Single-level discrete 2-D wavelet transform

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

Single-level inverse discrete 2-D wavelet transform

```matlab
1 % Computes 1-level discrete haar 2-D wavelet transform
2 [A, H, V, D] = dwt2(X, 'haar');
3 % Computes 1-level inverse discrete haar 2-D wavelet transform
4 X = idwt2(A, H, V, D, 'haar');
```

9 Artificial Neural Networks

Feed-Forward Neural Networks

Example: XOR gate
% Create feed-forward network
% Parameters:
% 1- Max and min of the input values (2 inputs)
% 2- Number of layers 2 layers, 1st (hidden) 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)