1 Follow instructions to display different image types and reproduce their details using MATLAB

1.1 Objectives

Understand MATLAB user interface and learn basic MATLAB commands for matrices and image operations.

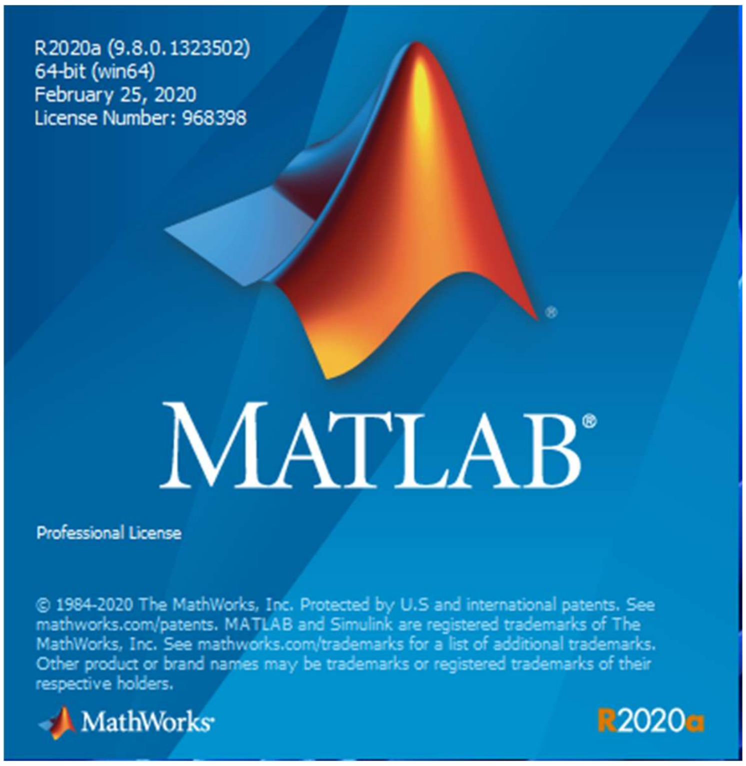
1.2 Pre-Lab

Learning to used MATLAB interface, MATLAB Help, vectors, and matrices in MATLAB.

1.2.1 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar **mathematical** notation. Typical uses include the following:

- Math and computation
- Algorithm development
- Data acquisition
- Modelling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building.



R2020a (9.8.0.1323502)
64-bit (win64)
February 25, 2020
License Number: 968398

MATLAB®

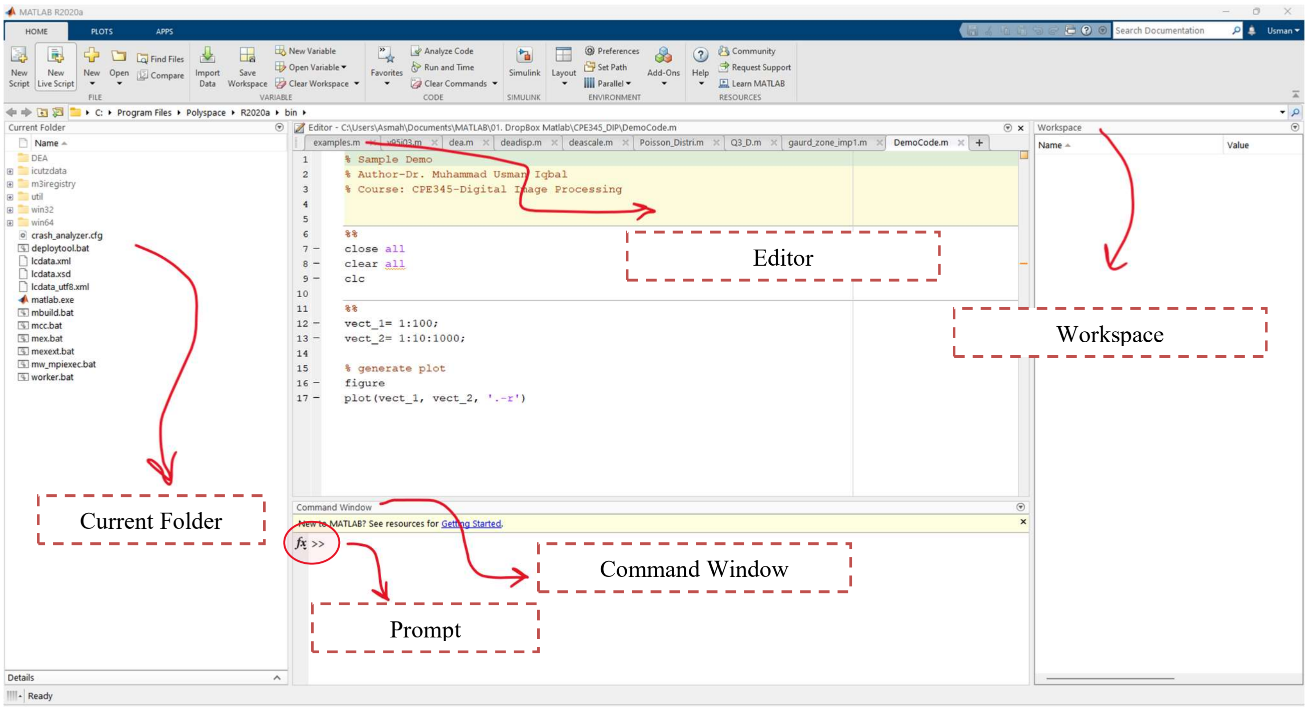
Professional License

© 1984-2020 The MathWorks, Inc. Protected by U.S and international patents. See mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

MathWorks® R2020a

Figure 1.1: Use a version of MATLAB 2018 or above.

1.2.2 MATLAB User Interface



```
% Sample Demo
% Author-Dr. Muhammad Usman Iqbal
% Course: CPE345-Digital Image Processing


%%
close all
clear all
clc

%%
vect_1= 1:100;
vect_2= 1:10:1000;

% generate plot
figure
plot(vect_1, vect_2, '.-r')
```

Pre-Lab Task 1

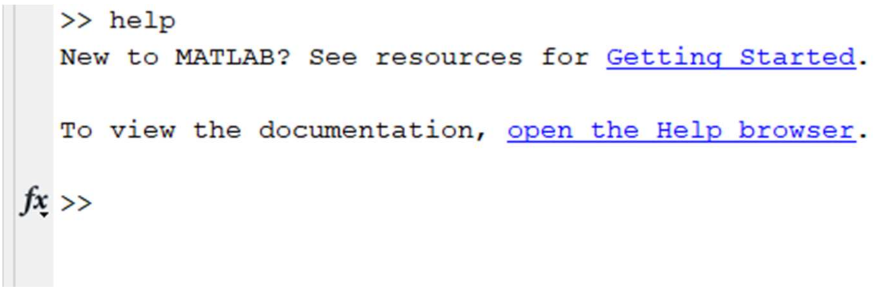
Practice the tools listed in Table 1.

Table 1.1: List of MATLAB Desktop Tools

Tool	Description
Array Editor	View and edit array contents.
Command History Window	View a log of statements entered in the **Command Window**; search for previously executed statements, copy them, and re-execute them.
Command Window	Run MATLAB statements.
Current Folder Browser	View and manipulate files in the current folder.
Current Folder Field	Shows the path leading to the current folder.
Editors	Editor/Debugger and Live Editor (explained in the text).
Figure Windows	Display, modify, annotate, and print MATLAB graphics.
File Comparisons	View detailed differences between two files.
Help Browser	View and search product documentation.
Profiler	Measure execution time of MATLAB functions and lines; count how many times code lines are executed.
Start Button	Run product tools and access product documentation.
Workspace Browser	View and modify contents of the workspace.

1.2.3 How to use MATLAB help

Online help is available from the MATLAB prompt (a double arrow), both generally (listing all available commands). In the text that follows, any line that starts with two greater than signs (>>) is used to denote the MATLAB command line (also known as prompt). This is where you enter your commands.



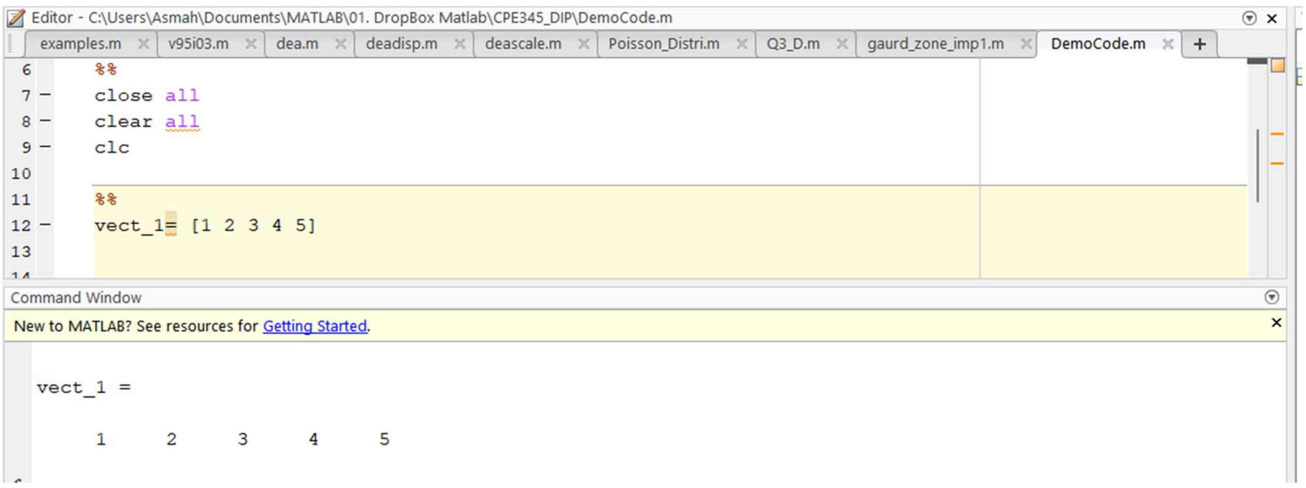
```
>> help
New to MATLAB? See resources for Getting Started.

To view the documentation, open the Help browser.

fx >>
```

1.2.4 Vectors and Matrices in MATLAB

In this section a brief overview of vectors and Matrices generation in MATLAB is described through examples.

1.2.4.1 Defining a Vector

Almost all of MATLAB's basic commands revolve around the use of vectors. A vector is defined by placing a sequence of numbers within square braces:



Editor - C:\Users\Asmah\Documents\MATLAB\01. DropBox Matlab\CPE345_DIP\DemoCode.m

examples.m | v95i03.m | dea.m | deadisp.m | deascale.m | Poisson_Distri.m | Q3_D.m | gaurd_zone_imp1.m | DemoCode.m

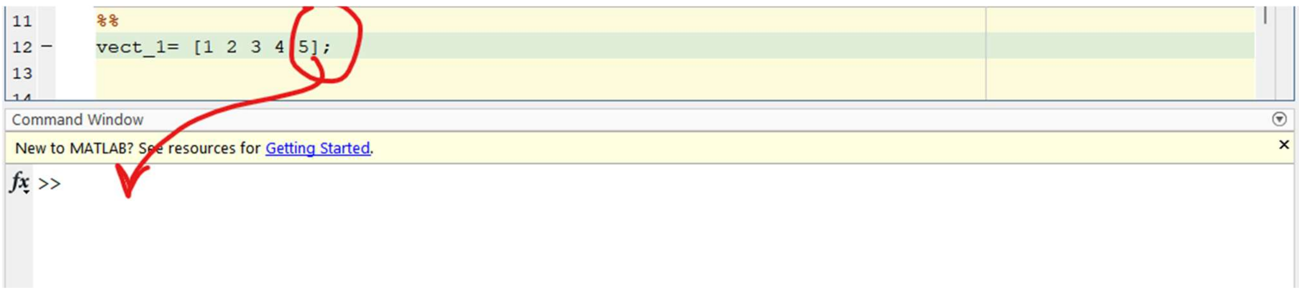
```
%%
close all
clear all
clc

%%
vect_1= [1 2 3 4 5]
```

Command Window

New to MATLAB? See resources for Getting Started.

```
vect_1 =

     1     2     3     4     5
```

This creates a row vector which has the label "vect_1". Note that MATLAB printed out a copy of the vector after you hit the enter key. If you do not want to print out the result, put a semi-colon at the end of the line:



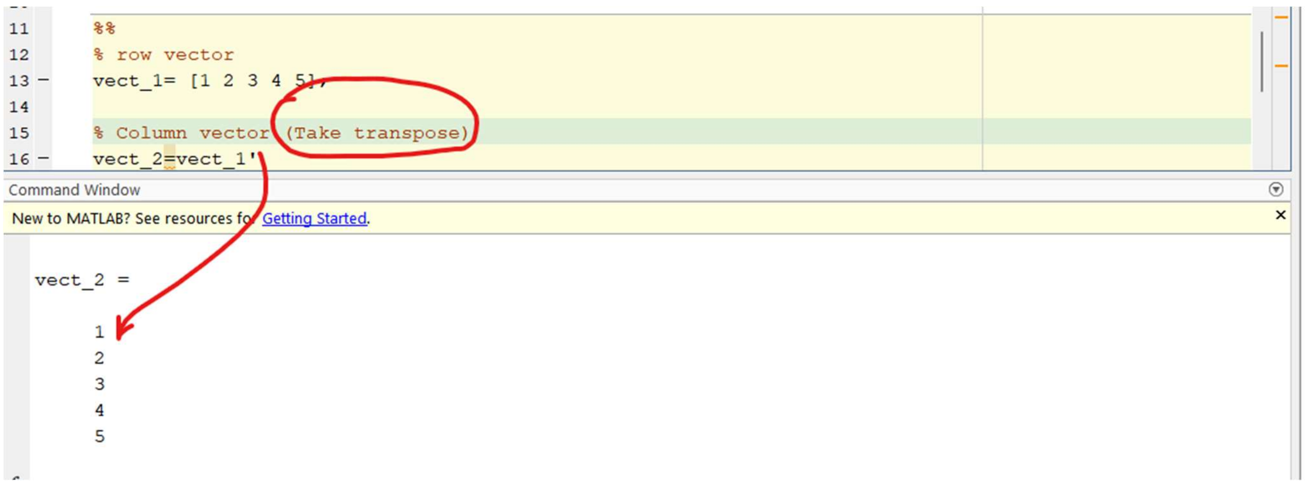
```
%%
vect_1= [1 2 3 4 5];
```

Command Window

New to MATLAB? See resources for Getting Started.

```
fx >>
```

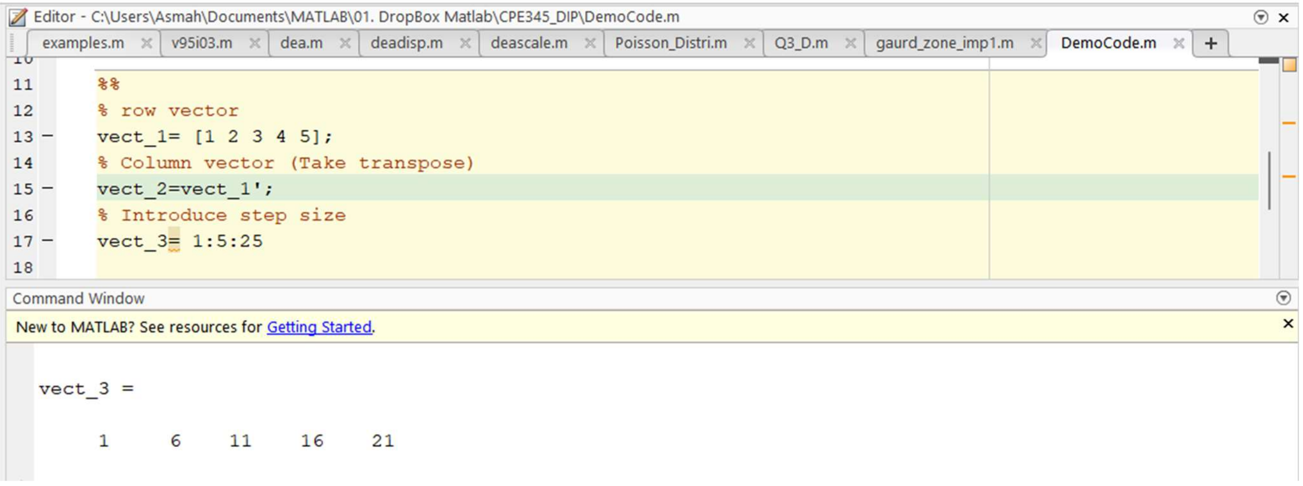
Notice, though, that this always creates a row vector. If you want to create a column vector you need to take the transpose of a row vector. The transpose is defined using an apostrophe ("' ")



```
%%
% row vector
vect_1= [1 2 3 4 5];

% Column vector (Take transpose)
vect_2=vect_1'
```

```
vect_2 =

     1
     2
     3
     4
     5
```

If you wish to use an increment other than one that you have to define the start number, the value of the increment, and the last number. For example, to define a vector that starts with 2 and ends in 4 with steps of .25 you enter the following:



Editor - C:\Users\Asmah\Documents\MATLAB\01. DropBox Matlab\CPE345_DIP\DemoCode.m
examples.m
v95i03.m
dea.m
deadisp.m
deascale.m
Poisson_Distri.m
Q3_D.m
gaurd_zone_imp1.m
DemoCode.m

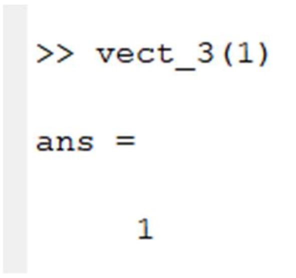
```
%%
% row vector
vect_1= [1 2 3 4 5];
% Column vector (Take transpose)
vect_2=vect_1';
% Introduce step size
vect_3= 1:5:25
```

Command Window
New to MATLAB? See resources for Getting Started.

```
vect_3 =

     1     6    11    16    21
```

Accessing elements within a vector

You can view individual entries in this vector. For example, to view the first entry just type in the following:

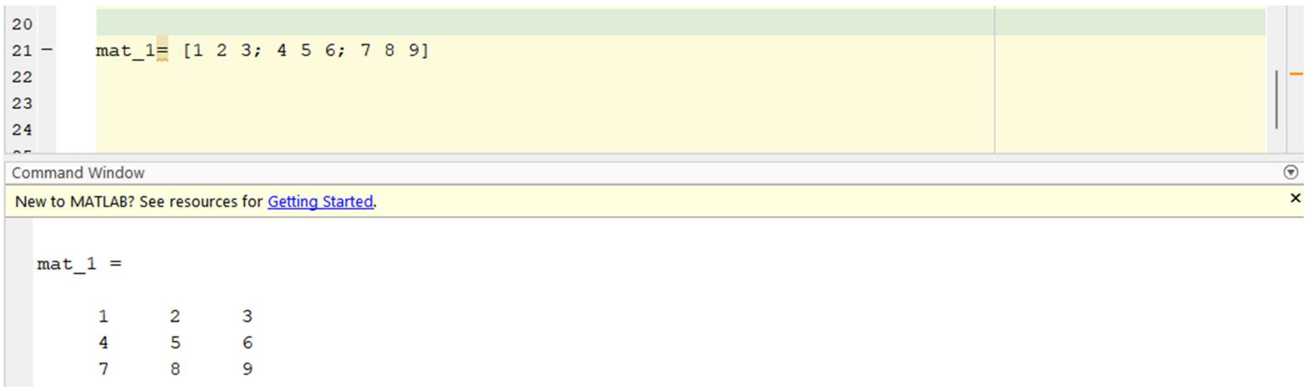


```
>> vect_3(1)

ans =

     1
```

This command prints out entry 1 in the vector. Also notice that a new variable called **ans** has been created. Any time you perform an action that does not include an assignment MATLAB will put the label ans on the result.

1.2.4.2 Defining Matrices

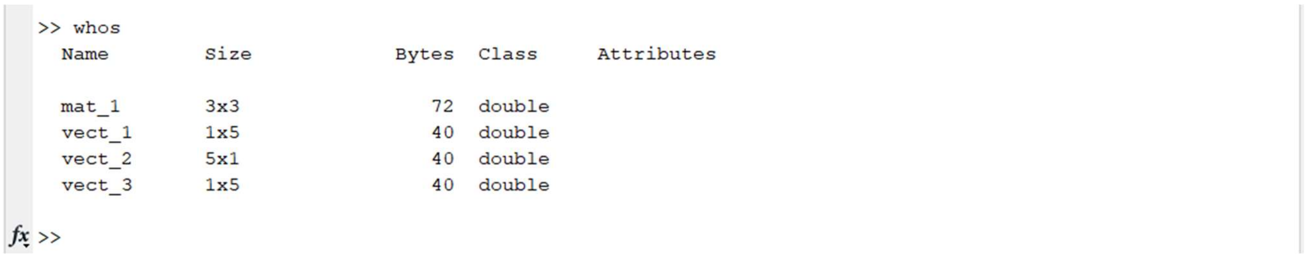
Defining a matrix is similar to defining a **vector**. To define a matrix, you can treat it like a column of row vectors (note that the spaces are required!).



```
mat_1= [1 2 3; 4 5 6; 7 8 9]
```

```
mat_1 =

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

If you have been putting in variables through this and the tutorial on **vectors**, then you probably have a lot of variables defined. If you lose track of what variables you have defined, the `whos` command will let you know all of the variables you have in your work space.



```
>> whos
  Name        Size            Bytes  Class     Attributes

  mat_1       3x3                72  double
  vect_1      1x5                40  double
  vect_2      5x1                40  double
  vect_3      1x5                40  double

fx >>
```

You can work with different parts of a matrix, just as you can with vectors. Again, you have to be careful to make sure that the operation is legal.

Pre-Lab Task 2

Write and execute .m file with name "Lab_1_PreLab_RegistrationNumber.m" to perform the following tasks:

a. Generate a row vector of the size 1x51.

b. Generate a column vector of the size 1000x1000.

c. Generate two matrices' zeroes and ones of size 1000x1000.

true(M, N)
false(M, N)
magic(M)
rand(M,N)

1.2.4.3 Digital Image Processing Fundamentals

The basic data structure in MATLAB is the array, an ordered set of real or complex elements. This object is naturally suited to the representation of images, real-valued ordered sets of color or intensity data. MATLAB stores most images as two-dimensional arrays (i.e., matrices), in which each element of the matrix corresponds to a single pixel in the displayed image. (Pixel is derived from picture element and usually denotes a single dot on a computer display.)

For example, an image composed of 200 rows and 300 columns of different colored dots would be stored in MATLAB as a 200-by-300 matrix. Some images, such as RGB, require a three-dimensional array, where the first plane in the third dimension represents the red pixel intensities, the second plane represents the green pixel intensities, and the third plane represents the blue pixel intensities. This convention makes working with images in MATLAB similar to working with any other type of matrix data and makes the full power of MATLAB available for image processing applications.

1.3 In-Lab Tasks

In-Lab Task 1

Using the MATLAB built-in functions, generate matrices of zeros and ones of the size $M = 100\ and\ N = 1000$. Use MATLAB function `imshow` to display each image separately in figure window. Your task is to generate an image like this or any other pattern of black and white strips.

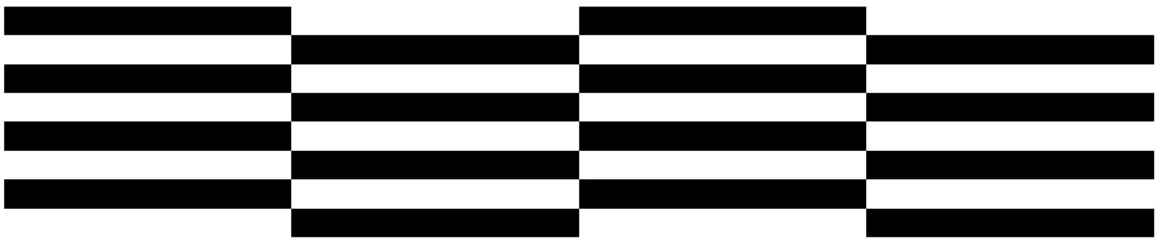


Figure 1.2: Black and White Strips

Use MATLAB Help for the following functions.

zeros
ones
imshow
figure

1.3.1 Reading an Image

To import an image from any supported graphics image file format, in any of the supported bit depths, use the `imread` function.

```
A = imread(filename,fmt)
```

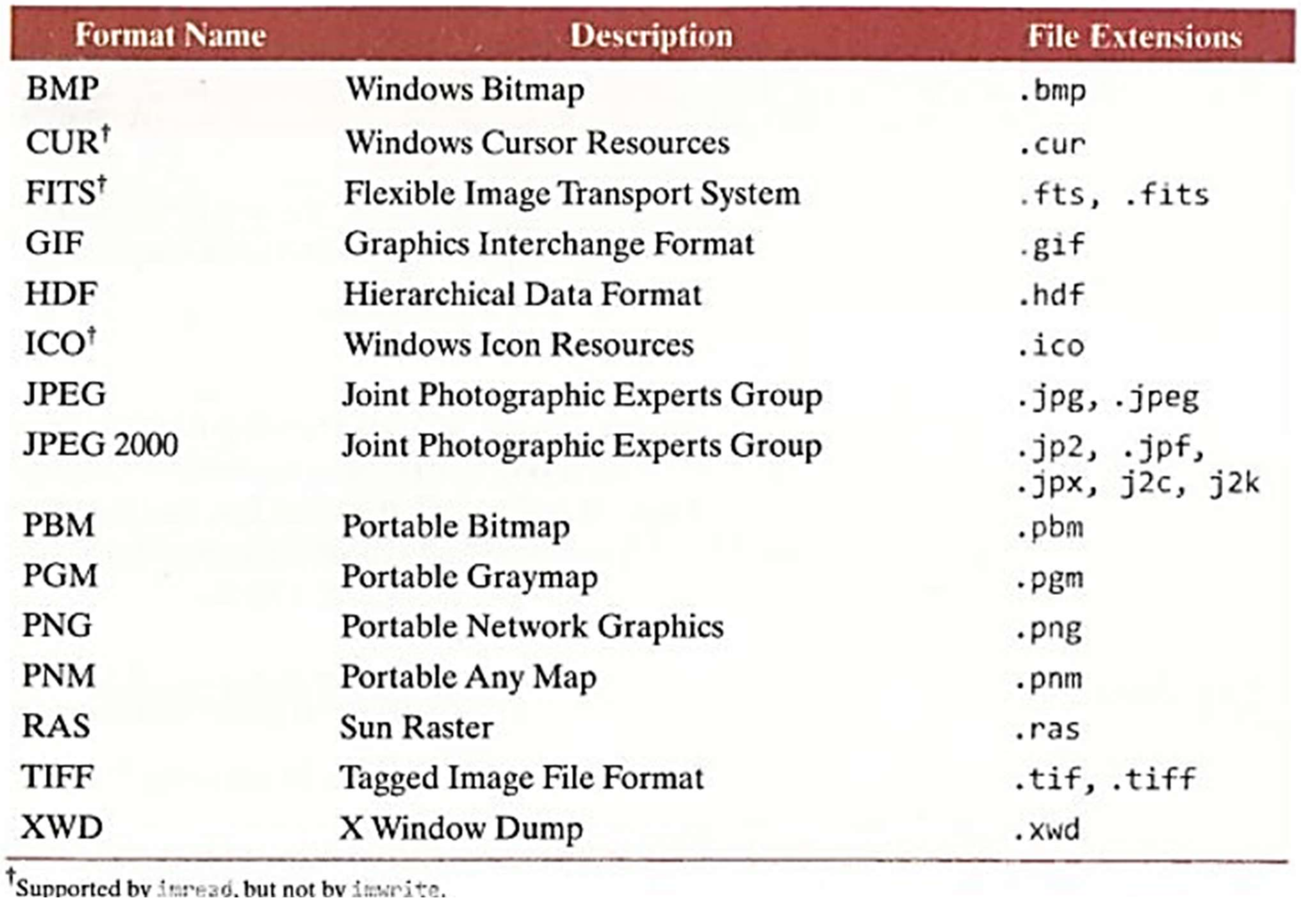
reads a greyscale or color image from the file specified by the string filename, where the string `fmt` specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname of the location on your system.

Example:

```
A = imread('chestxray.jpg');
```

This reads the image from the JPEG file "chestxray,jpg" into image array A. Note the use of single quotes (‘ ‘) to delimit the string file name.

Table 1.2: Types of Image Formats supported by MATLAB



Format Name	Description	File Extensions
BMP	Windows Bitmap	.bmp
CUR†	Windows Cursor Resources	.cur
FITS†	Flexible Image Transport System	.fts, .fits
GIF	Graphics Interchange Format	.gif
HDF	Hierarchical Data Format	.hdf
ICO†	Windows Icon Resources	.ico
JPEG	Joint Photographic Experts Group	.jpg, .jpeg
JPEG 2000	Joint Photographic Experts Group	.jp2, .jpf, .jpx, j2c, j2k
PBM	Portable Bitmap	.pbm
PGM	Portable Graymap	.pgm
PNG	Portable Network Graphics	.png
PNM	Portable Any Map	.pnm
RAS	Sun Raster	.ras
TIFF	Tagged Image File Format	.tif, .tiff
XWD	X Window Dump	.xwd

†Supported by imread, but not by imwrite.

1.3.2 Size of the Image

After reading an image in MATLAB, its size can be determined using the MATLAB function 'size'.

```
% Determine the size of the image
 size(A)

 % assign image size to the variable for future use
 [M N]= size(A)

 % Alternatively, you can use
 s=size(A)
 M=s(1)
 N=s(2)
```

1.3.3 Display an Image

To display images, use the `imshow` function.

```
imshow(X)
```

Example

```
imshow(A);
```

Now, Use the syntax

```
imshow(A,[low high]);
```

It displays as black all values less than or equal to low, and as white all values greater than or equal to high. The values in between are displayed as intermediate intensity values.

Finally, the syntax

imshow(A,[]);

It sets variable low to the minimum value of array A and high to its maximum value. This form of `imshow` is useful for displaying images that have a low dynamic range or that have positive and negative values.

1.3.4 Writing Image Data

To write image to graphics file `imwrite` is used imwrite(A,filename,fmt)

Example

imwrite(A,'Test.tif');

Function `imwrite` writes the image as a TIFF file because it recognizes the ".tif"extension in the filename.Alternatively, the desired format can be specified explicitly with a third input argument. This syntax is useful when the desired file does not use one oftherecognized file extensions. For example, the following command writes f toa TIFF file called "Test".

imwrite(A,'Test','tif');

How to get no. of rows and columns of image

Function `size` gives the rows and columns dimension of image.

[r,c]=size(A)

r = no of Rows

c = no of columns

Sine and cosine waves plotting

Write a MATLAB code to plot sine and cosine waves with proper titles, labels and annotations, also use different markers and colour for both waves.

Fundamentals of image processing

Write a MATLAB code that perform the following operations.

1. Ask the user to enter the name of the image file.
2. Read the image file.
3. Store the file with a different format and name, the name and format should also be chosen by the user.