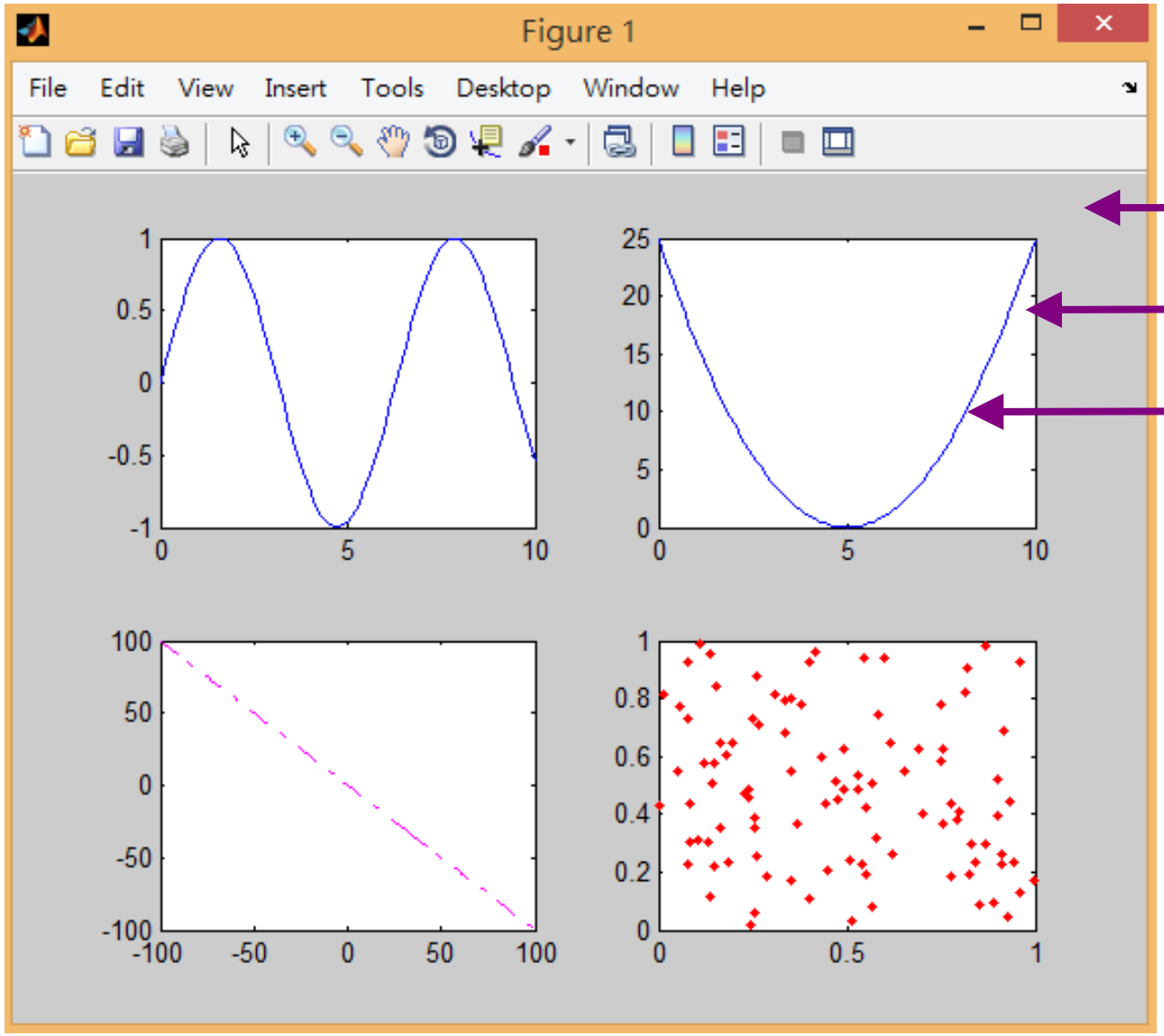
Unit 2: Data Visualization: Plots and Images

Graph Objects

Basic concepts: Three levels of graph objects in MATLAB:

- **Figure**: Each figure appears in a separate figure window. All other graph objects have to appear in figures.
- **Axes**: An axes specifies a particular coordinate system for drawing the data.
  - The display of an axes can be in 2-D (default) or 3-D.
- **Plots**: A "plot" is a way of displaying some data, such as a curve. Each plot belongs to an axes.

Graph Objects

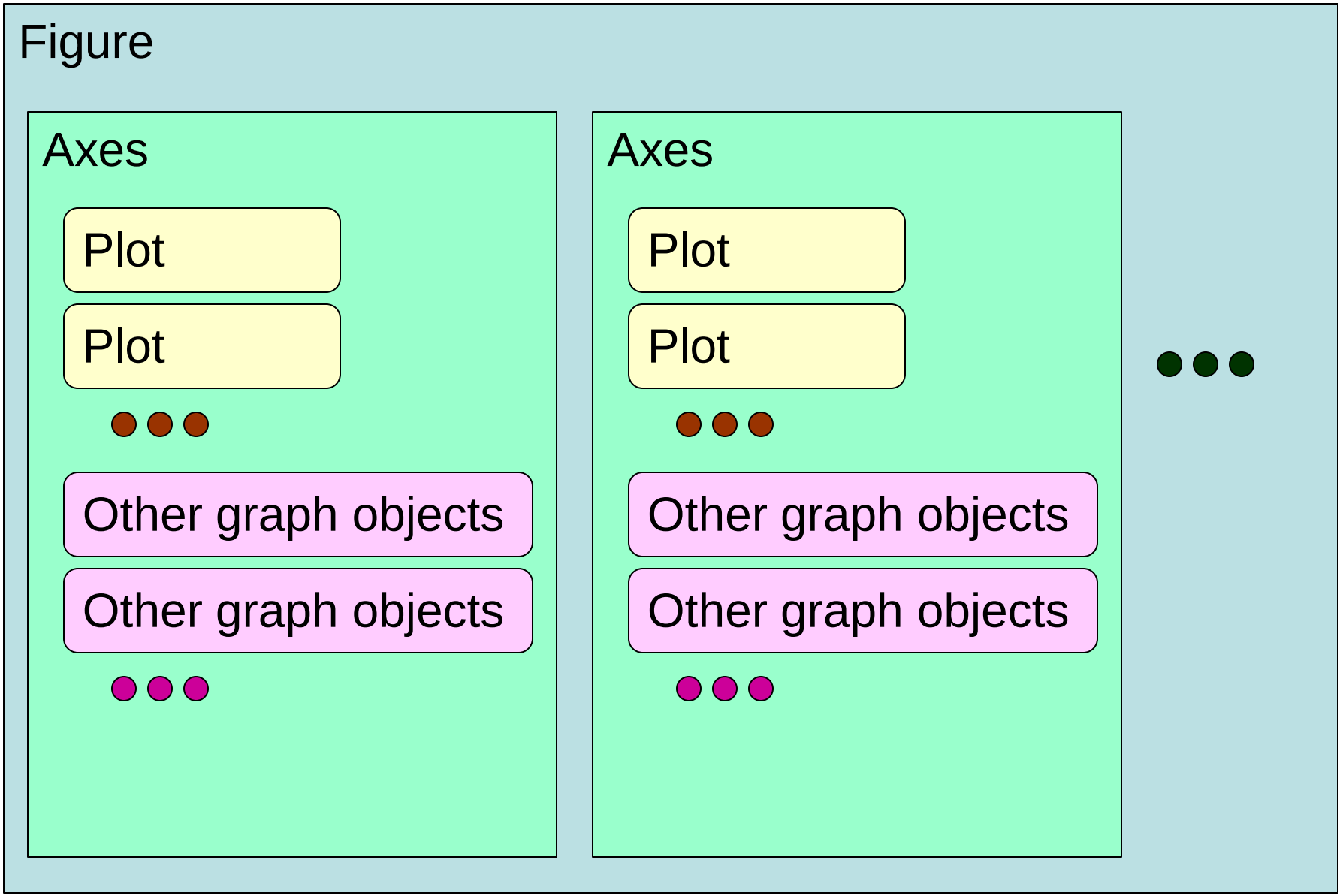


Figure

Axes

Plot (many types)

Graph Objects



Graph Objects

Generating and accessing graph objects:

- Each object has a **handle**, returned when created explicitly (such as by functions **`figure`**, **`axes`**, **`plot`**, etc.).
- When not specified, graph operations are applied to the **current** (most recently used) **figure** and **axes**.
  - Functions **`gcf`** and **`gca`** return the handles to the **current figure** and **current graph**, respectively.
  - A **container object** is automatically created for an operation if none exists. (Example: A figure and an axes are created if you just call **`plot`**.)
  - A call to **`figure(n)`**, where **`n`** is a positive integer, can create a new figure or set an existing figure with handle **`n`** as the current figure.

Multiple Axes in a Figure

- Specifically call **`axes`** for each axes:
  - Manually controlled position/size for each axes. The positions/sizes are relative to the containing figure.
- Call function **`subplot`** before drawing operations:
  - Automatically controlled position/size for each axes.

Basic 2-D Plots

- Basic form: `plot(X,Y)`
  - Here `X` and `Y` are vectors of the same length.
  - If `X` is omitted (called with syntax `plot(Y)`), MATLAB generates `X` using `1:length(Y)`.
  - This function creates what is called a **line plot** (the default plot type) in MATLAB.

Basic 2-D Plots

We can set basic plot properties within the call to **`plot`**:

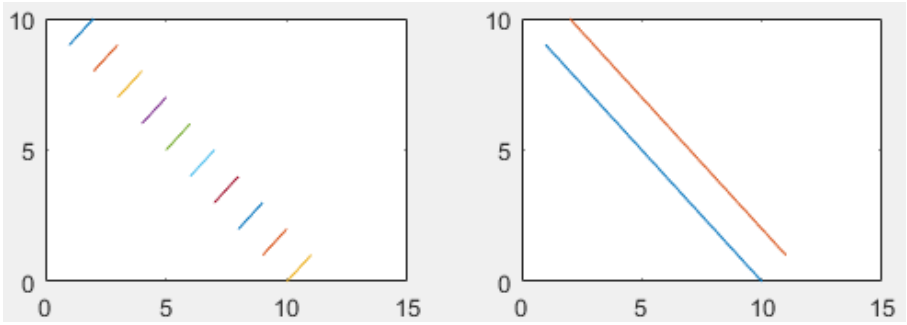
- Example: **`plot(X,Y,'o-m')`**
- Setting basic colors
- Setting line types
- Setting marker types
- Draw scatter plots by specifying marker types but no line types.
- The use of name-value pairs of properties:
  - Examples: **`'linewidth'`**, **`'color'`**, …
  - Many more; check MATLAB documentation
  - Specifying RGB colors

Multiple 2-D Plots in One Axes

Drawing multiple plots on the same axes:

- Method #1: Use **`X`** and **`Y`** that are arrays of the same size.
  - Each corresponding pair of column vectors in **`X`** and **`Y`** will become a line plot.
  - Line colors are assigned automatically if not specified.
  - Example codes:

```
x=1:10; y=10-x;
plot([x;x+1],[y;y+1]);
plot([x;x+1]',[y;y+1]');
```



- Method #2: Use **`hold on`** for the axes:
  - **`hold on/off`**: Whether old contents are retained when drawing new plots.

Axis Properties of 2-D Plots

Controlling the axes:

- Setting axis ranges:
  - **`axis([xmin xmax ymin ymax])`**
  - **`axis auto/manual/tight`**
- Axis/box visibility:
  - **`axis on/off`**
  - **`box on/off`** (the outside box)
- Aspect ratio: **`axis normal/equal/square`**
- Direction (where the origin is): **`axis xy/ij`**

Adding Text in 2-D Plots

Marking the axes:

- Marking on **x** and **y** axis:
  - Functions **`xlabel`** and **`ylabel`**:
  - Use **`set(gca, …)`** with name-value pairs:
    - Properties: **`'xtick'`**, **`'xticklabel'`**, **`'ytick'`**, **`'yticklabel'`**, …
- Function **`title`**:
- Function **`legend`**: For drawing the legend
- Function **`text`**: For drawing texts
  - Properties: **`'fontname'`**, **`'fontweight'`**, **`'fontsize'`**, **`'color'`**, **`'verticalalignment'`**, **`'horizontalalignment'`**, …

Graph Object Properties

- The various graph objects have many properties that can be queried and set.
  - Too many for us to list them; check the documentation.
- Many properties can be set with name-value pairs when creating a graph object.
- To query the properties of an existing object:
  - `get(handle, name-value pairs …)`
- To set the properties of an existing object:
  - `set(handle, name-value pairs …)`

Additional 2-D Plot Types

More on **x** and **y** axis:

- Log-scale plots:
  - Function **`loglog`**:
  - Functions **`semilogx`** and **`semilogy`**:
  - These functions are used in place of **`plot`**.
- Two **y** axis on the left and right sides; there are actually two **axes** overlapped on each other:
  - Function: **`plotyy(X1,Y1,X2,Y2)`**

Additional 2-D Plot Types

- For discrete numerical data:
  - Functions **`scatter`**, **`stem`** and **`stair`**:
- Plot in polar coordinates: Function **`polar`**
- Additional plot types:
  - **`pie`**
  - **`bar`**, **`barh`**
  - **`area`**
- Histogram plots:
  - **`hist`** (more in later lectures)
  - **`rose`** (angular histogram)

2-D Contour Plots

- Purpose: To display a function **`Z=f(X,Y)`** in 2-D
- Function: **`contour(X,Y,Z)`**
  - Basic form: **`contour(X,Y,Z)`**
  - Specifying the **`Z`** values to draw the contour lines: **`contour(X,Y,Z,v)`**, where **`v`** is a monotonically increasing vector.
  - Specifying the number of contour lines: **`contour(X,Y,Z,n)`**, where **`n`** is an integer.
  - Getting returned handles: **`[C,h]=contour(…)`**

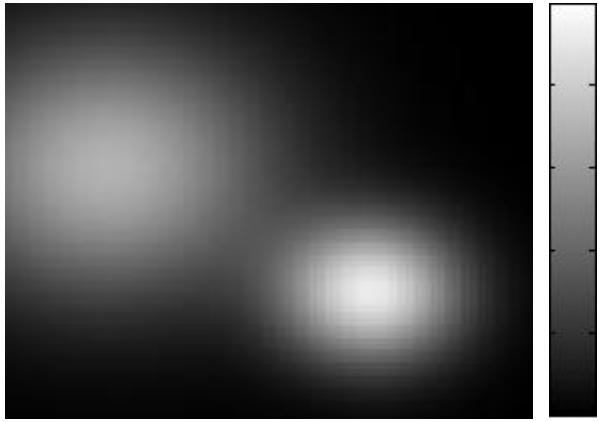
2-D Contour Plots

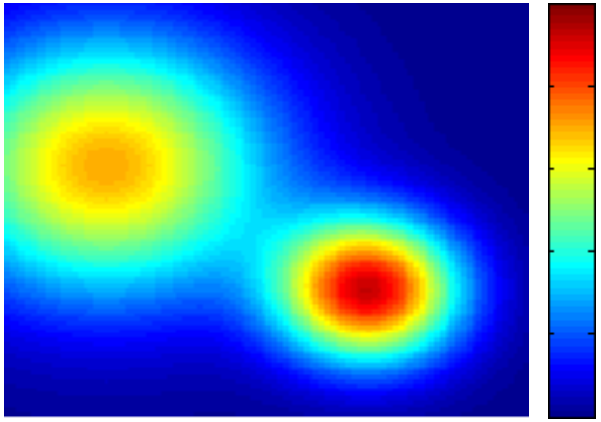
Additional features:

- Showing contour labels: Function **`clabel`**
  - Basic form: **`clabel(C)`** or **`clabel(C,h)`**
  - Labeling specific contours only:
  - Specifying text properties with name-value pairs:
  - Using returned text handles to change text properties (including the text string):
- Filled contour plots: **`contourf(X,Y,Z)`**
  - The use of **colormaps**:

An Introduction to Colormaps

- A mapping from scalar values (positive integers) to color values.
- Used to enhance displays (also called **pseudocolor** processing).





- Colormaps in MATLAB are `Mx3` arrays, each row being a RGB color (values are 0-1).
- Specifying a colormap: `colormap(map)`
  - The specified colormap applies to the whole figure.

An Introduction to Colormaps

- Predefined colormaps in MATLAB:
- Getting a predefined colormap:
  - Example: `jet(32)`

Pseudocolor 2-D Plots

- Function **`imagesc`**:
  - Basic form: **`imagesc(Z)`**, where **`Z`** is a 2-D array.
  - With specified **`x`** and **`y`** values (in vectors): **`imagesc(x,y,Z)`**
  - With specified **`Z`** range: **`imagesc(…,[zmin zmax])`**. If not specified, the full range of data is used.
  - The full range of the current colormap is used (default: **`jet(64)`**).
- Showing the value-color correspondence: Function **`colorbar`** (The "color bar" is displayed in another axes).

Miscellaneous: Creating 2-D Grids

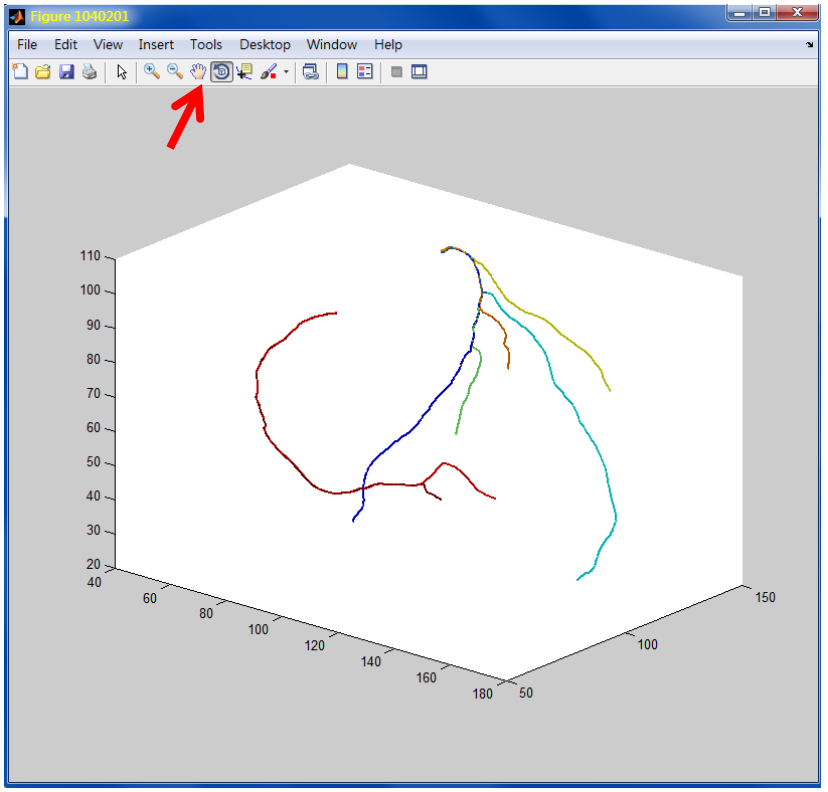
- We can plot 2-D functions in the form of $f(x,y)$ by sampling it on a 2-D grid..
- Function **`meshgrid`**:
  - Specific for 2-D.
  - Intuitive ordering of **x** and **y** coordinates: **`[X,Y]=meshgrid(x,y)`**, with **`x`** and **`y`** being vectors.
- Function **`ndgrid`**:
  - For 2-D or more dimensions.
  - Following MATLAB ordering of dimensions: **`[A,B]=ndgrid(a,b)`**, with **`a`** and **`b`** being vectors.
  - (For 2-D case), the created matrices are transposes of the matrices created with **`meshgrid`**.

3-D Plots

- Line series plots in 3-D: Function **`plot3`** (very similar to the use of **`plot`**).
  - **`plot3(x,y,z,…)`**.
- Scatter plots in 3-D: Function scatter3 (very similar to the use of **`scatter`**).
  - **`scatter3(x,y,z,…)`**.
  - One can specify the sizes or colors of individual markers: **`scatter3(x,y,z,s,c)`**.

Adjusting Views of 3-D Plots

- Interactive view adjustments:



- View adjustments in programs:
  - Adjusting viewpoints of 3-D plots: function **`view`**.
  - MATLAB provides a several functions with names **`cam*`** that controls the "camera" in 3-D views.

Additional 3-D Plots

- Some other plots types have 3-D versions that make them look better. Examples:
  - **`bar3`** and **`barh3`**
  - **`pie3`**
  - **`stem3`**
  - **`contour3`**

Mesh and Surface Plots

- Mesh and surface plots: Showing 2-D functions in 3-D views.
  - **`X`** and **`Y`** should be 2-D grids.
  - Each set of adjacent 2x2 grid points form a patch.
  - Function **`mesh(X,Y,Z,…)`** : Draws only the edges of the patches.
  - Function **`surf(X,Y,Z,…)`** : Draws the patch surfaces.

Image Basics

Basic types of images:

- Binary images, one bit per pixel. In MATLAB: **MxN** arrays of type **logical**. The images are black-and-white.
- Intensity images. In MATLAB: **MxN** arrays. Types can be integers (mostly **uint8**) or floating point (mostly **double**). The images are gray-scale.
- Color images. In MATLAB: **MxNx3** array. Types can be integers (mostly **uint8**) or floating point (mostly **double**).
  - The size of the third dimension is the number of **color planes**. Standard color images have 3 planes (R, G, B). Some image formats may use values other than 3, such as 4.

Image Basics

Basic types of images:

- For intensity and color images, when the data type is **uint8**, the values are from 0 to 255. When the data type is floating point, the values are from 0 to 1.
- Indexed images. In MATLAB: An **MxN** array of integers plus a color map.
  - Saves storage space for color images.
  - Makes re-coloring easy.
  - For many processing tasks (such as filtering), it is necessary to convert indexed images to regular (non-indexed) images first.

Loading Images

- Reading images from files: Function **`imread`**:
  - A non-indexed image is always put in an array of type **uint8** (most common) or **unit16**, or **logical** for 1-bit-per-pixel image files.
  - For indexed images, the color map can be retrieved together.

Displaying Images

- Function **`image`**:
  - Used in a way similar to **`plot`**; the image can be displayed as part of an axes using the coordinate system of that axes.
  - If displayed in a new axes, the **`YDir`** property of the axes is set to **`'reversed'`** (same as **`axis ij`**).
  - For indexed images, you need to supply the color map in the call.
- Function **`imshow`**:
  - Similar to **`image`**, but **`axis ij`**, **`axis equal`**, **`axis tight`** , and **`axis off`** are set automatically.

Writing Images

- Function **`imwrite`**:
  - Image file format can be automatically determined from the file name extension.
    - For some file formats, additional properties can be set with name-value pairs.
  - Images in floating-point arrays are always converted to type **uint8** (0→0 and 1→255) in the image files.
  - For indexed images, you need to supply the color map in the call.

Image Type Conversion

- Between data types: Functions **`im2double`**, **`im2uint8`**, **`im2uint16`**, **`im2single`**.
  - Automatic source type checking and scaling.
- Between color and gray-scale images: Function **`rgb2gray`**.
  - You can convert a gray-scale image **`A`** to a RGB image using **`cat(3,A,A,A)`** or **`repmat(A,[1 1 3])`**.
- To and from indexed images: Functions **`rgb2ind`**, **`ind2rgb`**, **`gray2ind`**, and **`ind2gray`**.
  - When converting to an index image, the color map can be supplied or generated automatically.