MATLAB:

A Comprehensive Introduction

Catherine Paverd

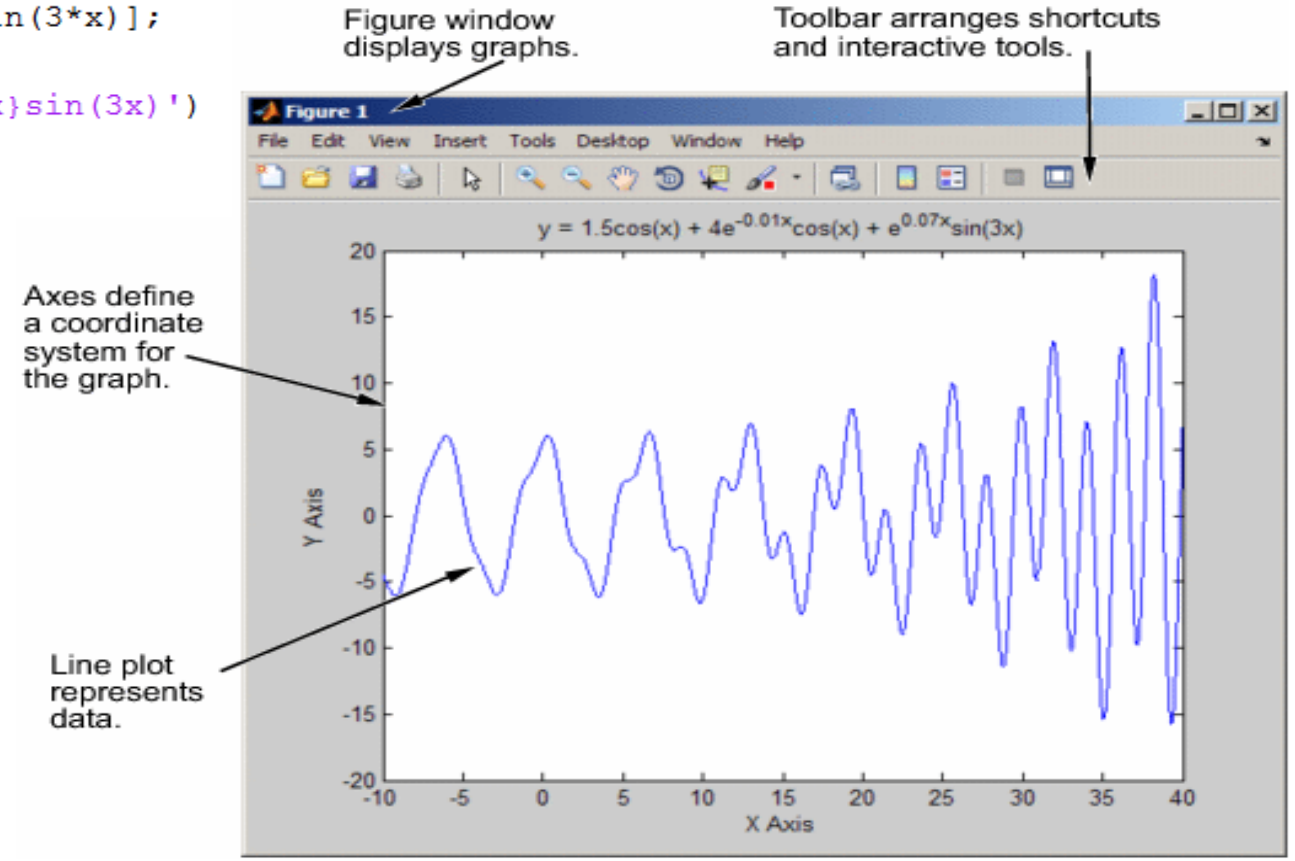
catherine.paverd@eng.ox.ac.uk

Graphics 1 – Figures, Axes and Graphs

- Some useful definitions:
  - **Figure**: a MATLAB window used for graphical display
  - **Axes**: 2D or 3D data space defined within a figure
  - **Plot**: any graphical display within a figure window
    - (figures can contain any number of plots and each plot is created within an axes)
  - **Graph**: a plot of *data* within an axes. Most plots are graphs.

Graph - Example

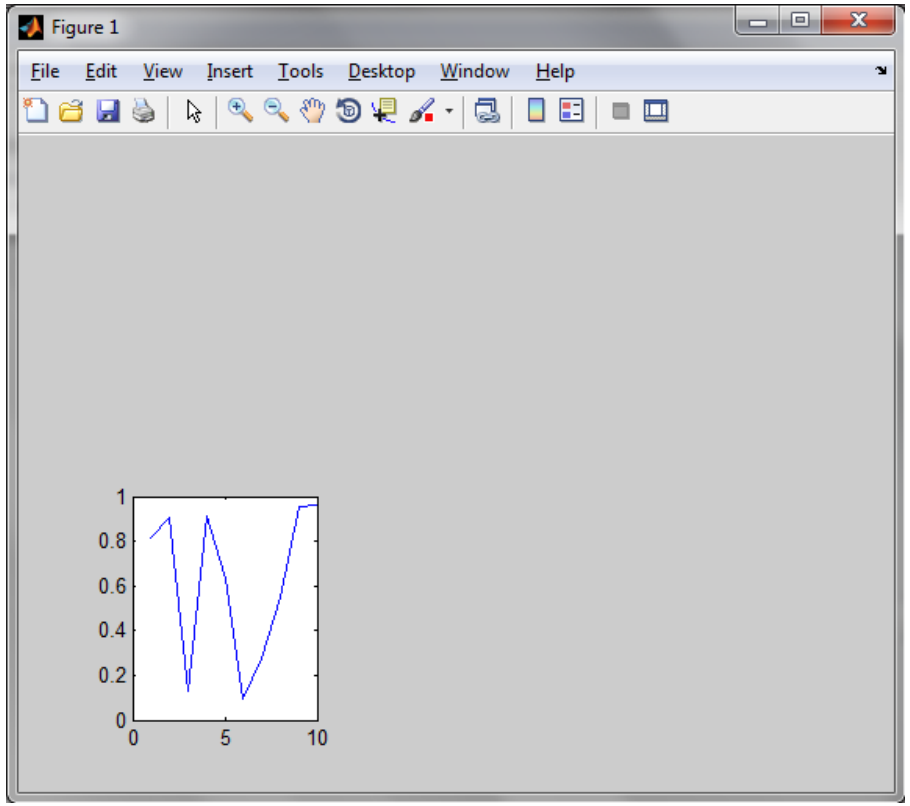
```
>> x = -10:.1:40;
>> y = [1.5*cos(x)+4*exp(-.01*x).*cos(x)+exp(.07*x).*sin(3*x)];
>> plot(x,y)
>> title ('y = 1.5cos(x) + 4e^{-0.01x}cos(x) + e^{0.07x}sin(3x)')
>> xlabel('X Axis')
>> ylabel('Y Axis')
```



Figures –Subplots

- **subplot** function: used to place multiple plots on a single figure in a regular tiled pattern.

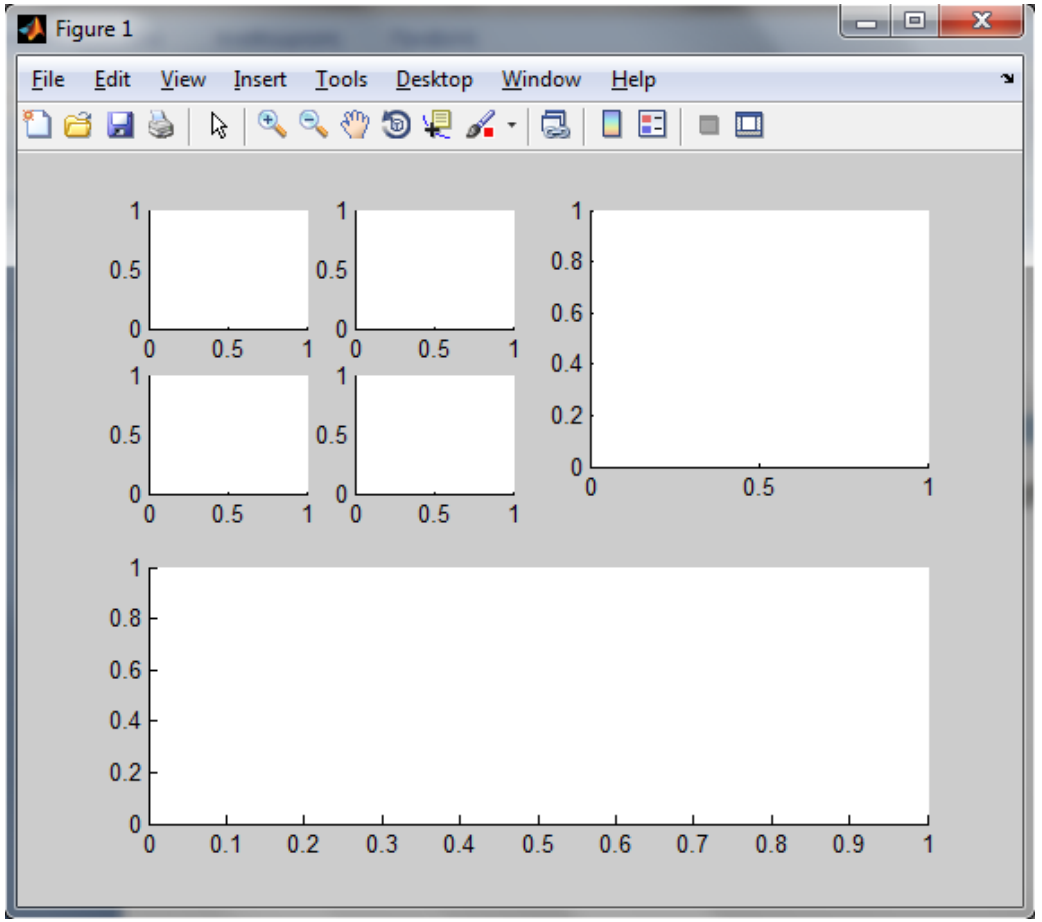
- *Example*:
  - subplot(2,3,4)
  - It defines a 2-by-3 grid on figure and opens or selects the 4th subplot counting along rows from the top left.
  - Now, a graphics command will output to this axes (e.g. plot(rand(1,10)).)



Figures –Subplots

- You can create multiple subplots if you avoid overlaps

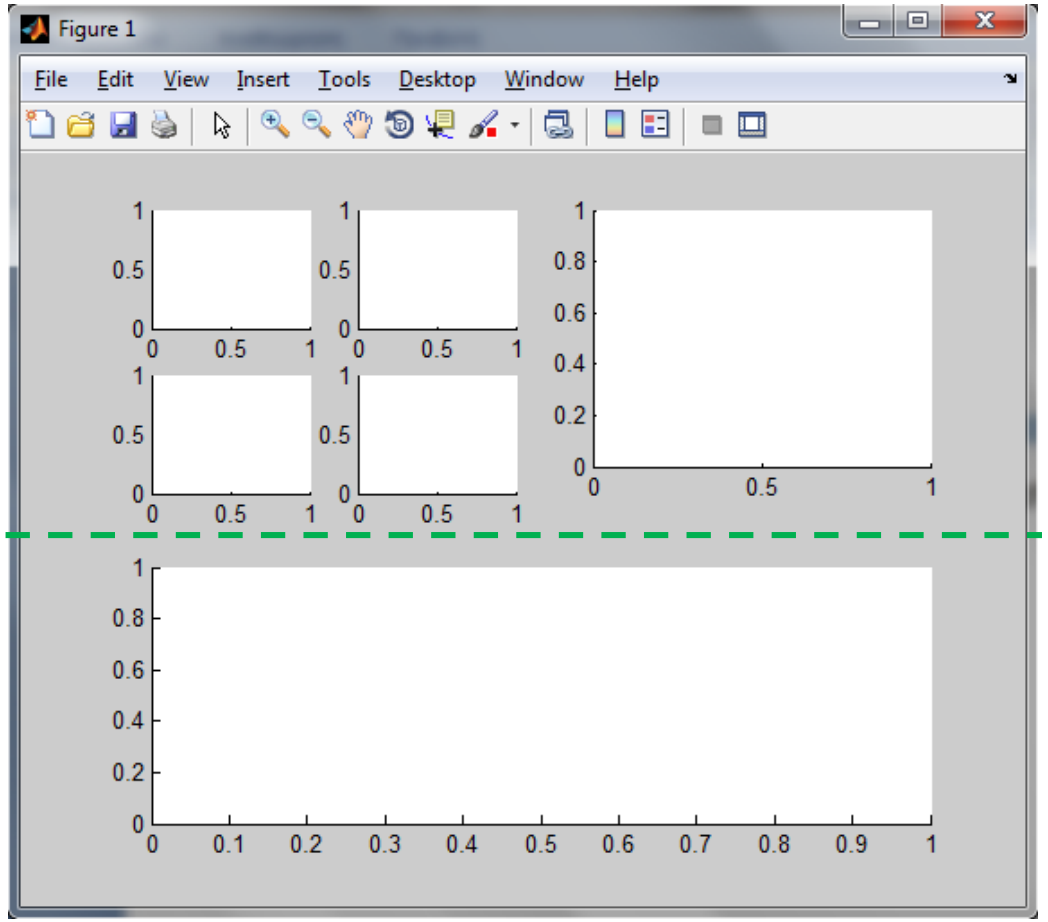
```
>> subplot(2,1,2)
>> subplot(2,2,2)
>> subplot(4,4,1)
>> subplot(4,4,2)
>> subplot(4,4,5)
>> subplot(4,4,6)
```



Figures –Subplots

- You can create multiple subplots if you avoid overlaps

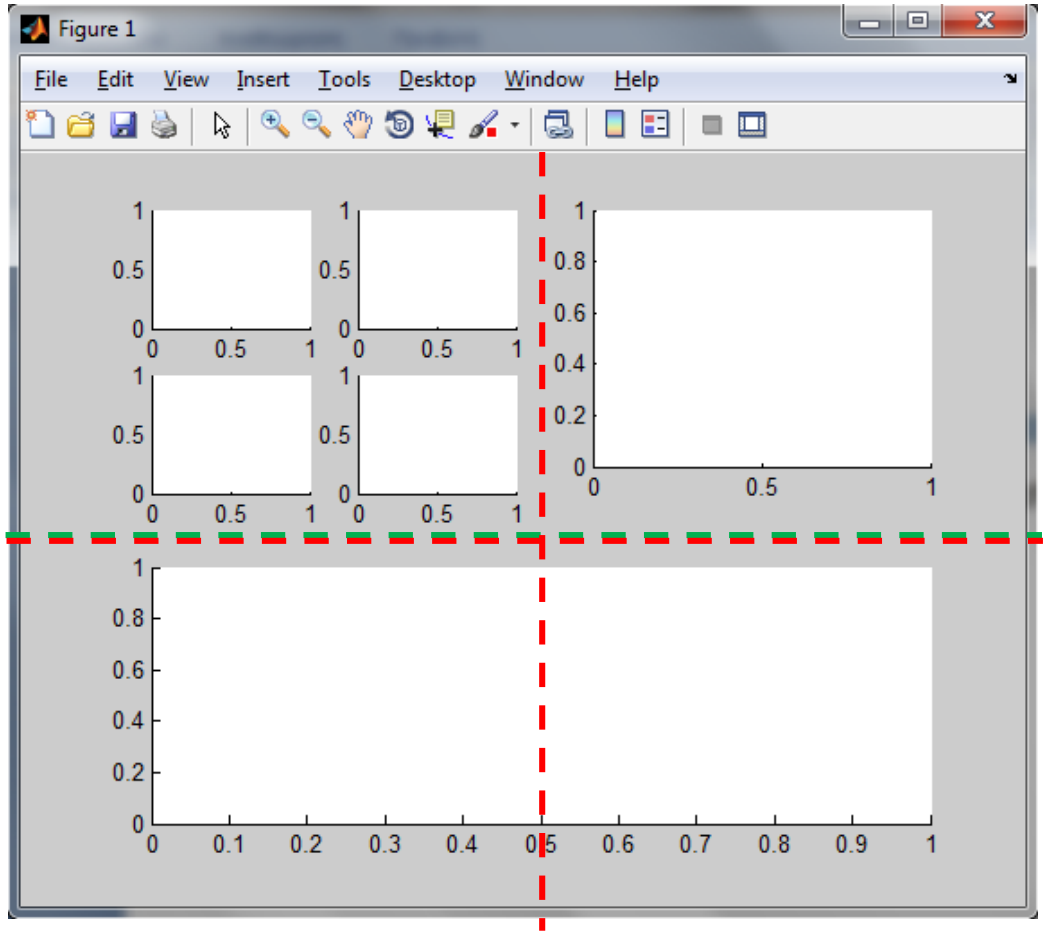
```
>> subplot(2,1,2)
>> subplot(2,2,2)
>> subplot(4,4,1)
>> subplot(4,4,2)
>> subplot(4,4,5)
>> subplot(4,4,6)
```



Figures –Subplots

- You can create multiple subplots if you avoid overlaps

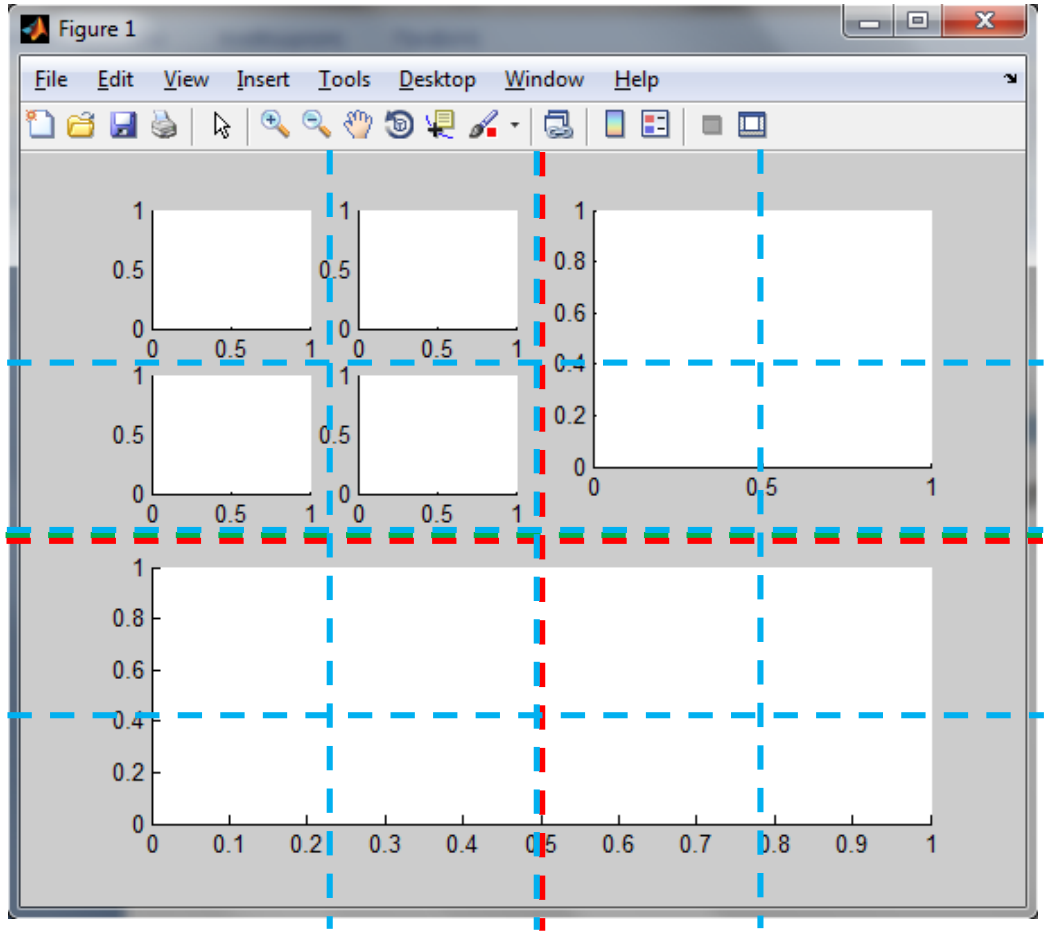
```
>> subplot(2,1,2)
>> subplot(2,2,2)
>> subplot(4,4,1)
>> subplot(4,4,2)
>> subplot(4,4,5)
>> subplot(4,4,6)
```



Figures –Subplots

- You can create multiple subplots if you avoid overlaps

```
>> subplot(2,1,2)
>> subplot(2,2,2)
>> subplot(4,4,1)
>> subplot(4,4,2)
>> subplot(4,4,5)
>> subplot(4,4,6)
```

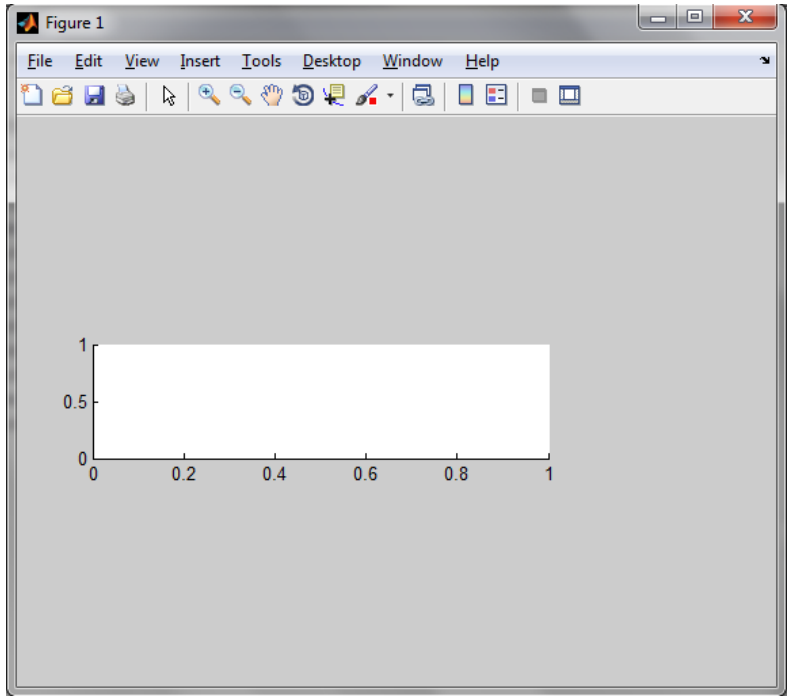


Figures –Subplots

- For complete flexibility, we can use the ‘position’ argument:
  - >> **subplot('Position', [left, bottom, width, height])**

  It creates an axes at the position specified by a 4-element numerical vector with normalised coordinates in (0,1).

  Example:
  >> figure
  >> subplot('position',[0.1 0.4 0.6 0.2])



Figures – Axes

axes: to create an axes and specify its properties.

>> **axes** : creates an axes object in the current figure using default properties.

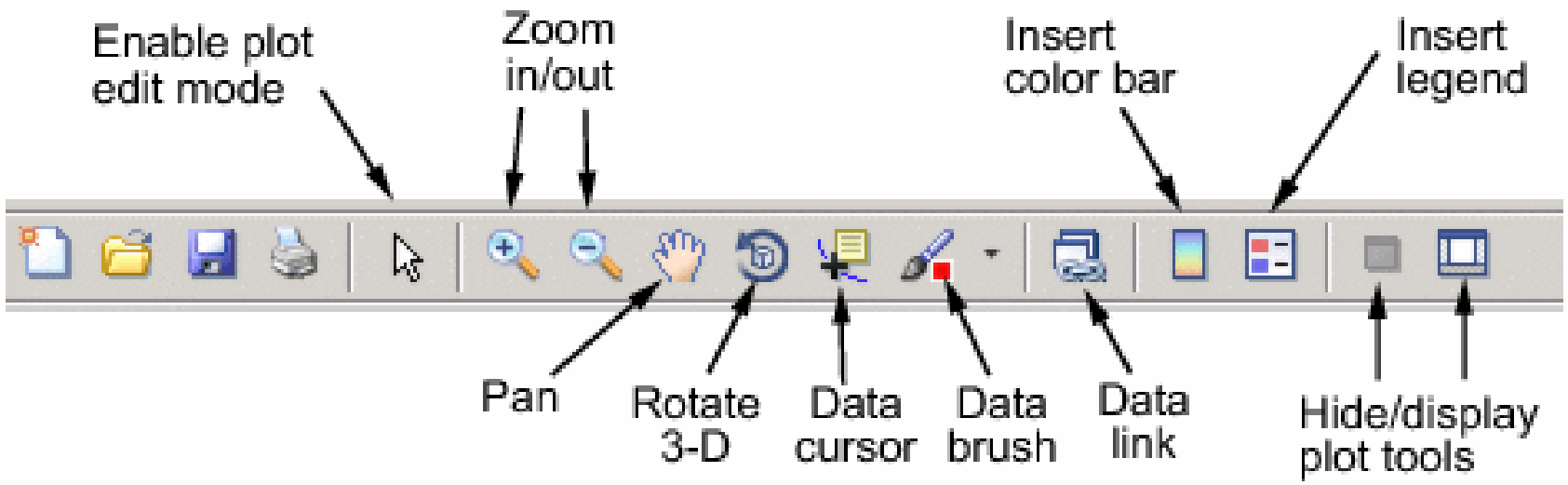
>> **axes('PropertyName',PropertyValue,...)** : creates an axes object having specific property values in name-value pairs.

axis: to manipulate commonly used axes properties, such as the axis limits:

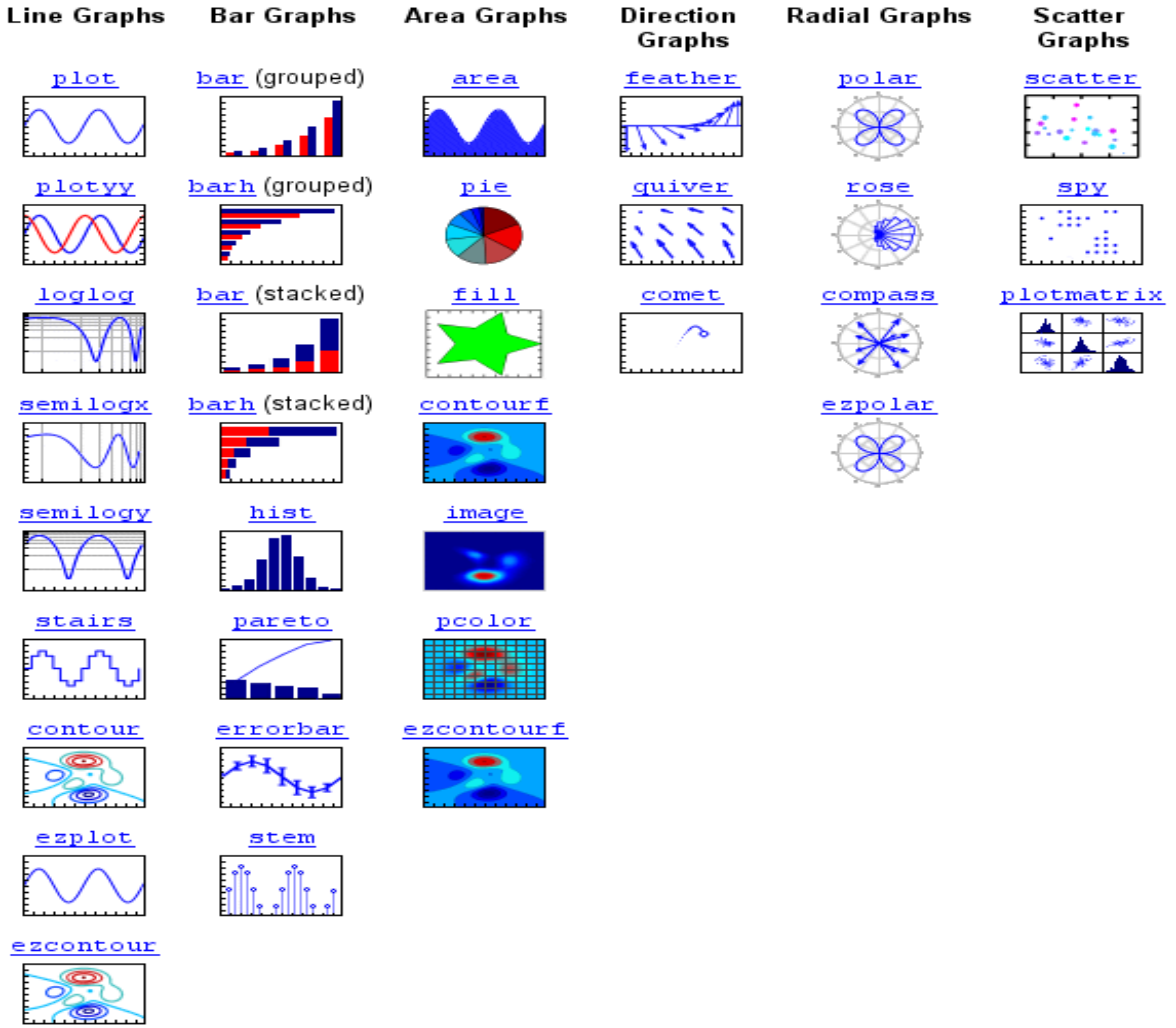
>> **axis([xmin, xmax, ymin, ymax])**

Figures – GUI Tools

Tools for manipulating and interacting with figure graphics are available in a toolbar at the top of the figure.



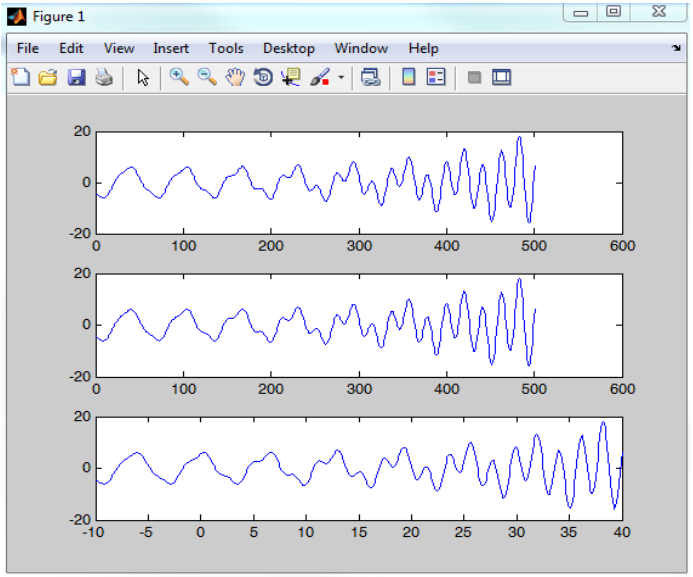
Plotting Functions – 2D



Line Graphs – plot()

- **plot**: the standard MATLAB line plotting function
- If **y** is a vector, then **plot(y)** produces a linear graph of y versus the index of its elements, **1:length(y)**.
- With two vector arguments, **plot(x,y)** plots **y** versus **x**

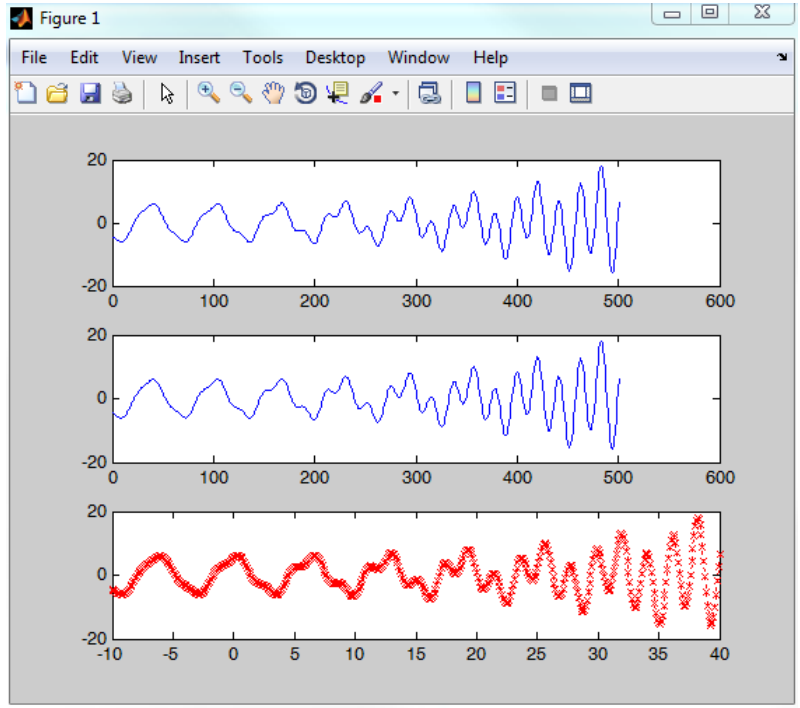
```
>> x = -10:.1:40;
y = [1.5*cos(x)+4*exp(-.01*x).*cos(x)+exp(.07*x).*sin(3*x)];
figure(1); clf;
subplot(3,1,1)
plot(y)
subplot(3,1,2)
plot(1:length(y),y)
subplot(3,1,3)
plot(x,y)
```



Line Graphs – plot() and LineSpec

- **LineSpec**: a way to change the properties of the line series plotted.
- *LineSpec string* consists of: **Line Style**, **Marker Symbol** and **Color** (in any order)

```
>> subplot(3,1,3)
plot(x,y,'r:x')
```



Line Graphs – plot() and LineSpec

```
Various line types, plot symbols and colors may be obtained with
PLOT(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:

      b     blue          .     point              -     solid
      g     green         o     circle             :     dotted
      r     red           x     x-mark             -.    dashdot
      c     cyan          +     plus               --    dashed
      m     magenta       *     star             (none)  no line
      y     yellow        s     square
      k     black         d     diamond
      w     white         v     triangle (down)
                          ^     triangle (up)
                          <     triangle (left)
                          >     triangle (right)
                          p     pentagram
                          h     hexagram

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus
at each data point; PLOT(X,Y,'bd') plots blue diamond at each data
point but does not draw any line.
```

Line Graphs – plot() and LineSpec

- Detailed property manipulation is available through **PropertyName**, **PropertyValue** pairs.
- *Example*:
```
>> plot(x,y,'r-s','MarkerEdgeColor','b','MarkerFaceColor','y')
```

- You can plot multiple x,y series using **plot(X1,Y1,LineSpec,...,Xn,Yn,LineSpec)**, but these statements can become rather long and unwieldy. An alternative is to use hold to accumulate plots on an axes.
- *Example*:
```
>> hold on
plot(x,y-20,'b-s','MarkerEdgeColor','g','MarkerFaceColor','r')
plot(x,y+20,'c-s')
```

- If one or both of an x, y pair is a matrix, multiple lines are plotted (one per column) using cycling, contrasting colours.

Bar Graphs – bar() and barh()

- Bar graphs display vector or matrix data as **vertical (bar)** or **horizontal (barh) bars**.

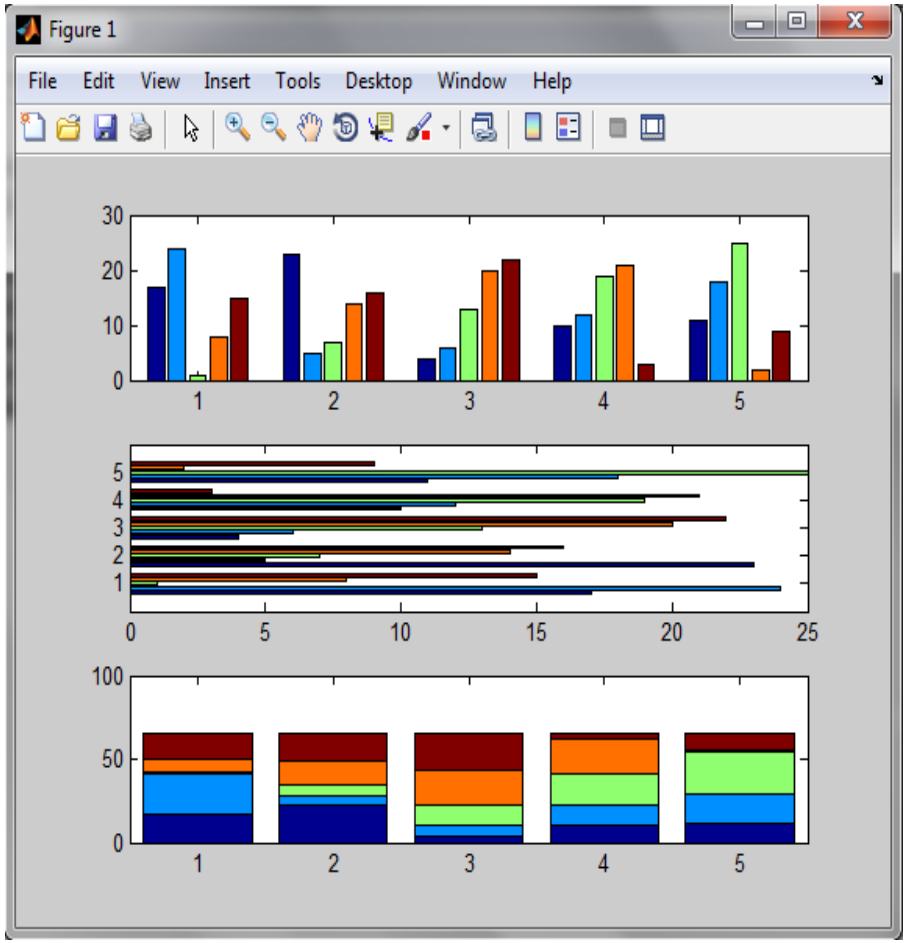
- **bar(Y)**:
  - draws one bar for each element in Y (if Y is a vector)
  - groups the bars produced by the elements in each row (if Y is a matrix)

- The x-axis scale ranges:
  - from 1 up to length(Y) when Y is a vector
  - from 1 up to number of rows when Y is a matrix

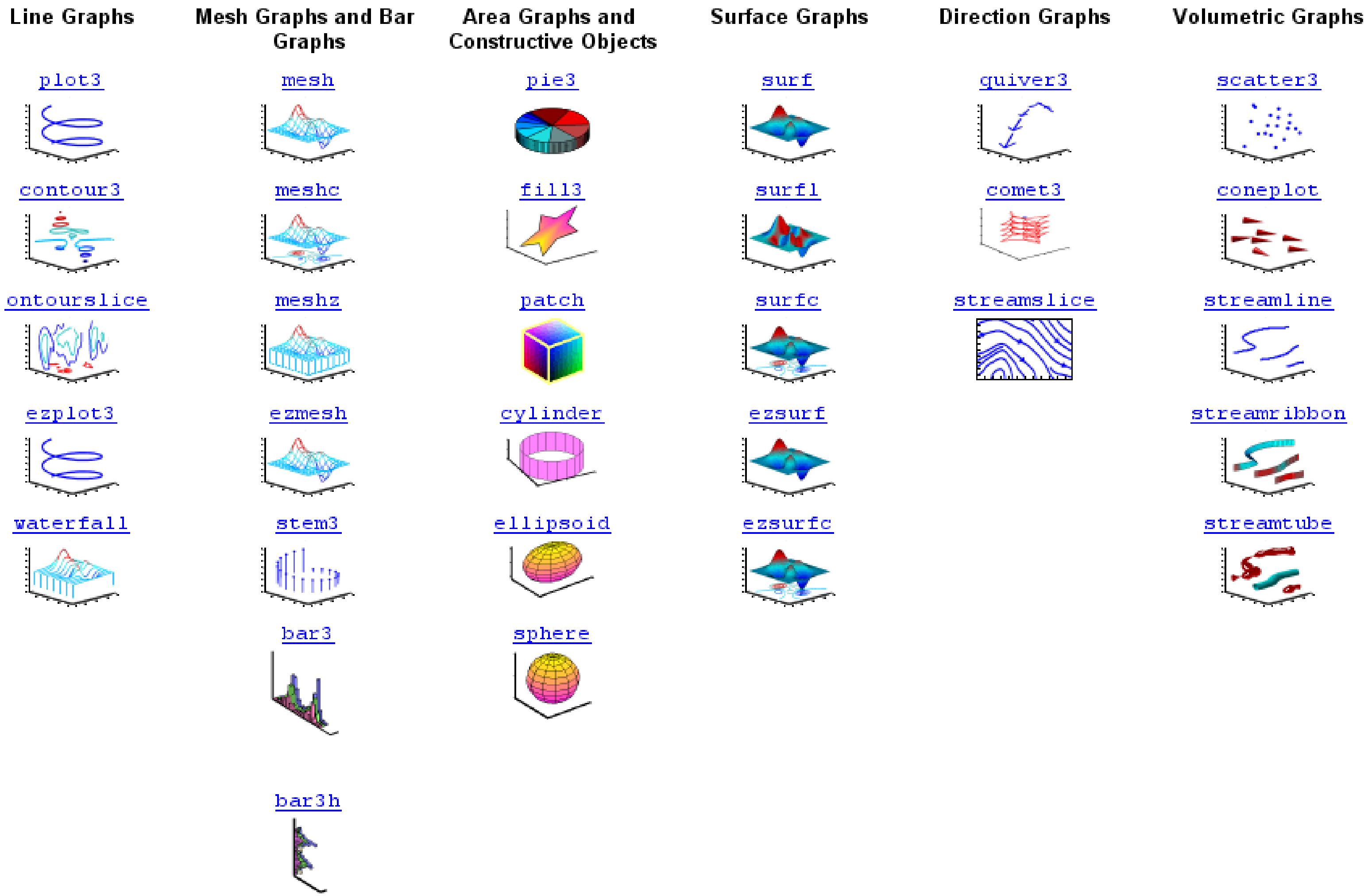
- **bar(X,Y)**: draws bars at locations specified by x.
- The ‘style’ argument specifies the choice between ‘grouped’ (default) and ‘stacked’ bars.

Bar Graphs – bar() and barh()

```
>> figure
>> subplot(3,1,1)
>> bar(magic(5))
>> subplot(3,1,2)
>> barh(magic(5))
>> subplot(3,1,3)
>> bar(magic(5),'stacked')
```



Plotting Functions – 3D



End of Part 1
Please start on Exercises 12 and 13

Part 2
Graphics 2 – Objects and Images

The Object Hierarchy

Objects are organised in a hierarchy in MATLAB, as shown in the figure below.

Graphics 2- Objects and Images

- Figures, axes and plots: examples of **graphics objects**

- **handle:**
  - a unique identifier for each instance of an object
  - it is returned by functions that create graphics objects
  - for the current figure and axes it is returned by **gcf** and **gca**
  - used to alter the properties of a graphics object
  - used to specify property values when a graphics object is created.

- *Example*:

```
>> h1 = figure
   h1 = 1

>> h2 = axes('Box','on','Color','y')
h2 = 173.0015

>> gcf
ans = 1

>> gca
ans = 173.0015
```

Objects, Handles and Properties

- Graphics objects properties allow us to alter figure appearance.

- Extensive range of options are available by searching in the help:
  - Figure Properties
  - Axes Properties
  - Lineseries (or Chart Line) Properties

Querying and Setting Property Values

- **Get** and **set**: two functions for fine-tuning graphics by querying and setting specific graphics object properties.
- **get(h)**: returns all properties of the graphics object identified by the handle h and their current values.
- **get(h, 'PropertyName')**: returns the value of thee named property
- **set(h)**: returns adjustable properties and possible values for object h.
- **set(h, 'PropertyName',PropertyValue,…)**: sets the named properties to the specified values.

Reading and Writing Image Files

- The major read and write functions for image files are **imread** and **imwrite**.
- **imfinfo**: returns information about an image file.
- **imformats**: displays a table listing the supported file formats

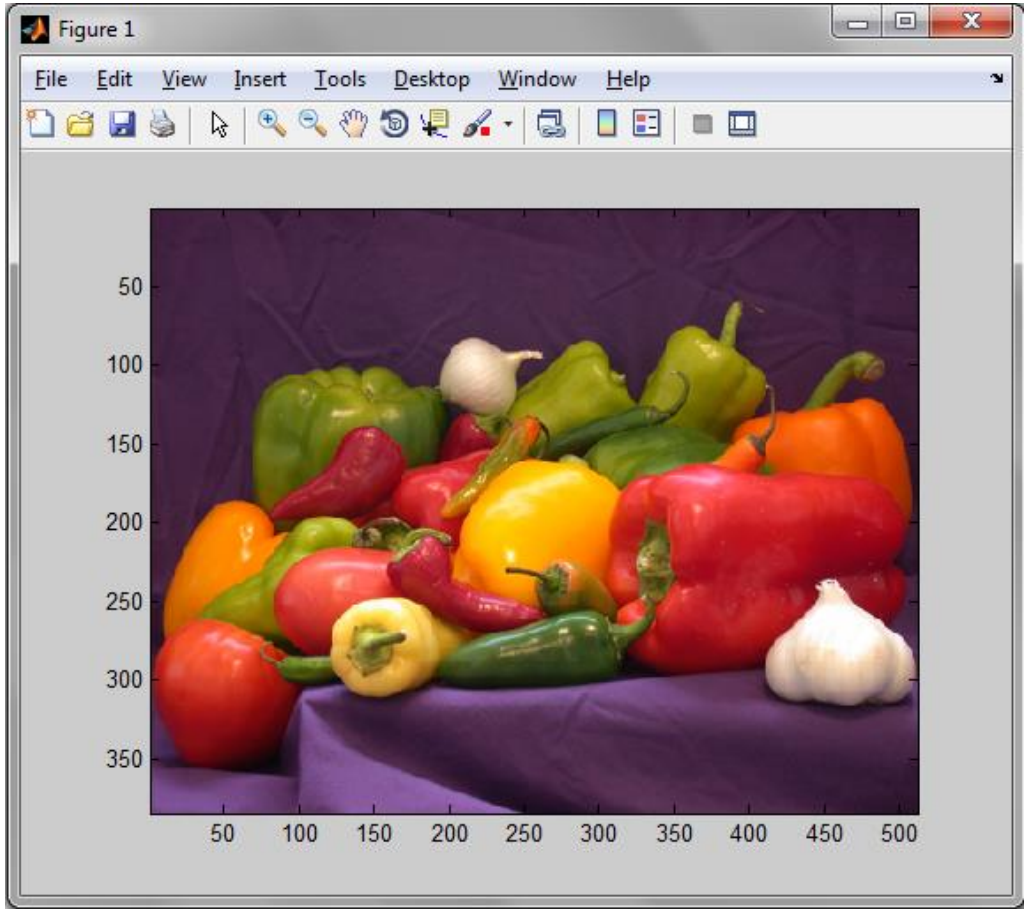
```
>> imformats

EXT       ISA     INFO        READ      WRITE     ALPHA  DESCRIPTION
---------------------------------------------------------------------------------------
bmp       isbmp   imbmpinfo   readbmp   writebmp  0      Windows Bitmap (BMP)
cur       iscur   imcurinfo   readcur             1      Windows Cursor resources (CUR)
fts fits  isfits  imfitsinfo  readfits            0      Flexible Image Transport System (FITS)
gif       isgif   imgifinfo   readgif   writegif  0      Graphics Interchange Format (GIF)
hdf       ishdf   imhdfinfo   readhdf   writehdf  0      Hierarchical Data Format (HDF)
ico       isico   imicoinfo   readico             1      Windows Icon resources (ICO)
j2c j2k   isjp2   imjp2info   readjp2   writej2c  0      JPEG 2000 (raw codestream)
jp2       isjp2   imjp2info   readjp2   writejp2  0      JPEG 2000 (Part 1)
jpf jpx   isjp2   imjp2info   readjp2             0      JPEG 2000 (Part 2)
jpg jpeg  isjpg   imjpginfo   readjpg   writejpg  0      Joint Photographic Experts Group (JPEG)
pbm       ispbm   impnminfo   readpnm   writepnm  0      Portable Bitmap (PBM)
pcx       ispcx   impcxinfo   readpcx   writepcx  0      Windows Paintbrush (PCX)
pgm       ispgm   impnminfo   readpnm   writepnm  0      Portable Graymap (PGM)
png       ispng   impnginfo   readpng   writepng  1      Portable Network Graphics (PNG)
pnm       ispnm   impnminfo   readpnm   writepnm  0      Portable Any Map (PNM)
ppm       isppm   impnminfo   readpnm   writepnm  0      Portable Pixmap (PPM)
ras       isras   imrasinfo   readras   writeras  1      Sun Raster (RAS)
tif tiff  istif   imtifinfo   readtif   writetif  0      Tagged Image File Format (TIFF)
xwd       isxwd   imxwdinfo   readxwd   writexwd  0      X Window Dump (XWD)
```

Displaying Images

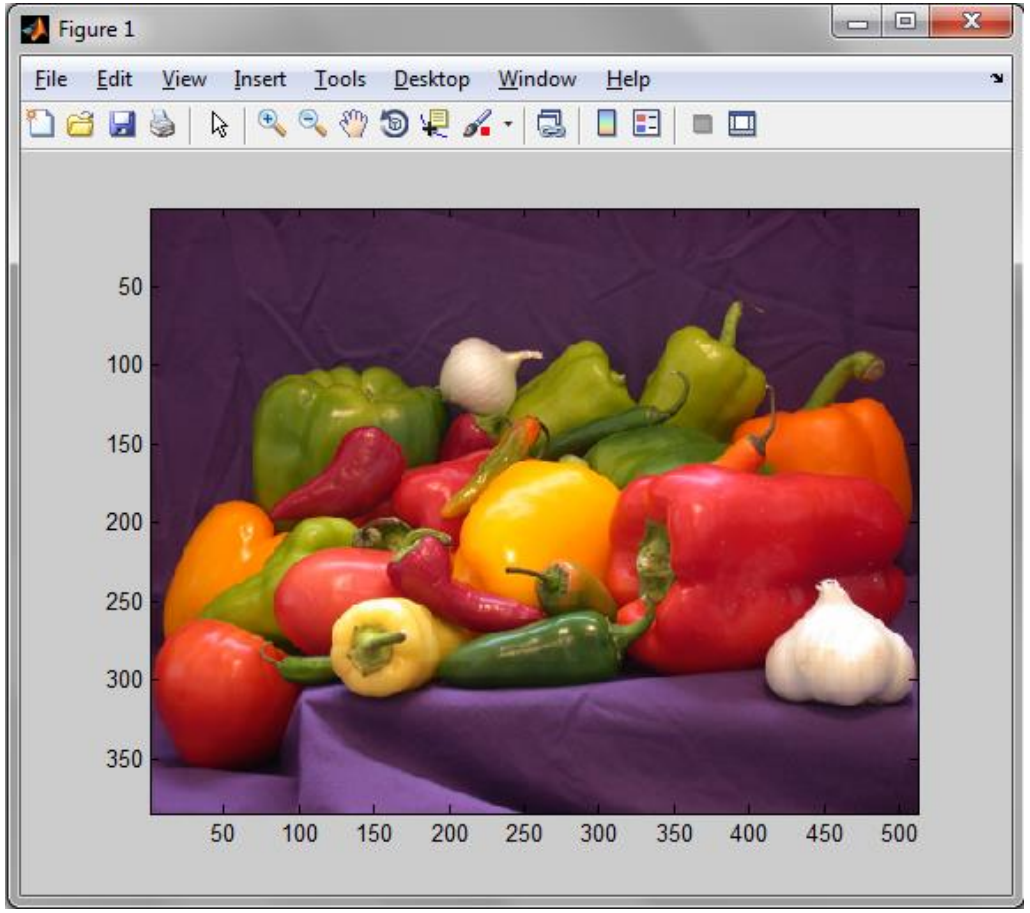
- To display an image, use **image** or **imagesc**.
- *Example*:

```
>> figure
>> peppers =
imread('peppers.png','png');
>> image(peppers)
```



Displaying Images

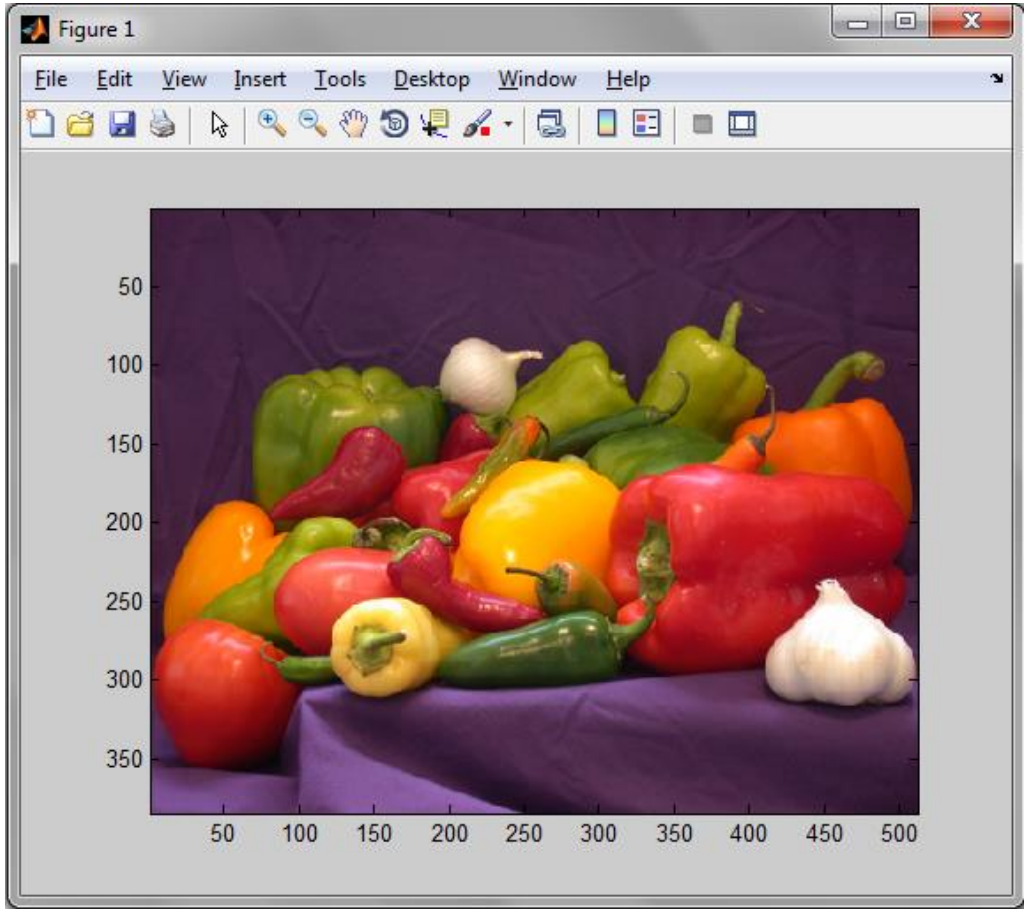
- To remove axis ticks we can use:
  >> axis off
- To correct aspect ratio for an image we can use:
  >> axis image



Displaying Images

To display an image with 1-to-1 mapping of matrix elements to screen pixels, we can resize the figure and axes:

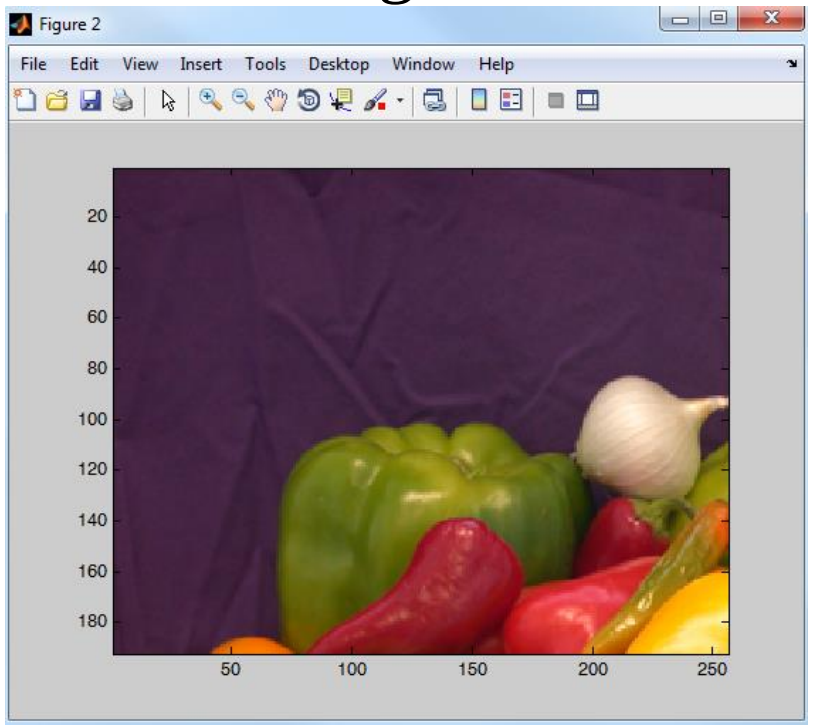
```
>> close all
[m,n,z] = size(peppers);
figure('Units','pixels','Position',[100 100 n m])
image(peppers); axis off;
set(gca,'Position',[0 0 1 1])
```



Displaying Images

Images can be cropped in MATLAB using sequential indexing:

```
>> [m,n,z] = size(peppers);
>> peppercrop = peppers(1:(m/2),1:(n/2),:);
figure; image(peppercrop)
```



End of Part 2
Please start on Exercises 14 and 15

Part 3
Debugging

Types of Errors

- **Typographical errors**: misspelled commands/variables (MATLAB error returned in some cases)
- **Syntax errors**: when calling a function, or using an operator (MATLAB error returned)
- **Array Errors:** incorrect indexing or operations
- **Algorithmic errors**: error in the process followed to produce the result (MATLAB error NOT returned)

Debugging in the Editor

- **Debugger with breakpoints**
- ***Try catch* statements**
- **Manual error finding**
  - Output
  - Automatic tests

```
try
   some code
catch
   error handling code
end
```

End of Part 3
Please start on Exercise 11