MATLAB has an outstanding set of graphic features and options. For example, it is very easy to plot a given data set or the results of computations. Analyzing mathematical equations with graphics is an efficient way of understanding and learning mathematics. Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ and $\mathbf{y} = [y_1, y_2, \ldots, y_n]$ be two vectors (arrays) of the same form (i.e., either row or column) and of the same length. The MATLAB command `plot(x,y)` plots vector $\mathbf{y}$ versus vector $\mathbf{x}$ and then joins the points (x_i, y_i), $i = 1, 2, \ldots n$, by straight line segments.

Exercise 1: Plot `y=[-3 1 -2 0 1 3]` versus `x=[-2 0 1.5 3 4 7]`.

Exercise 2: Plot the function $y_1 = \sin x$ on $[0, 2\pi]$ using 100 data points.
Solution:

```
x=linspace(0,2*pi,n); y1 = sin(x); plot(x, y1)
```

Comment: The `plot` function has different different forms depending on the input arguments. If y is a vector `plot(y)` produces a piecewise linear graph of the elements of y versus the index of the elements of y. If y is a complex vector, then `plot(y)` is equivalent to `plot(real(y),imag(y))`.

Adding titles, subtitles, axis labels, and legends
The following script will create two vectors, put a title and subtitle (using `title` and `subtitle`) on the plot, and add labels to the axes (using `xlabel` and `ylabel`).

```
x=0:0.1:5;
y=x.^2-2*x+cos(x);
plot(x,y)
title('y versus x plot') subtitle('Example')
xlabel('x-axis')
ylabel('y-axis')
```

The color of a single curve is, by default, blue, but other colors and styles are possible.

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 3 columns of the following table.

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

For example,

```
x=[-2 0 1.5 3 4 7];
y=[-3 1 -2 0 1 3];
plot(x,y,'r:*')
```

plots a red dotted (broken) line with an asterisk at each data point.

Related `plot` Features

`legend` creates a legend with descriptive labels for each plotted data series (type `help legend` to receive more information)

`figure` creates a new, empty Figure Window

`grid` displays grid lines on a plot

`axis([xmin xmax ymin ymax])` gives control over the end points of the axes. Also, you can force the two axes to have the same scale by `axis equal` or `axis image` and to have the same length by `axis square`

Specialized 2D plotting functions: `fplot`

- `fplot(fun,[xmin xmax])` plots the function `fun` on the interval `[xmin xmax]`.
- `fplot(fun,[xmin xmax],'LineSpec')` plots the function `fun` on the interval `[xmin xmax]` with the given line specification. For example,
  `fplot(@(x)x.^2,[-1 1],'r-')`
  plots the graph of $y = x^2$ in red.
- `fplot(x,y,[tmin tmax])` plots the parameterized curve with coordinates x(t), y(t) for t between [tmin, tmax]. For example,
  `fplot(@(t)cos(t),@(t)t.*sin(t),[0 2*pi])`
  plots the curve $x = \cos(t)$, $y = t\sin(t)$ for $t \in [0, 2\pi]$.
  'LineSpec' can be added as well.

Most of the next built-in functions display the same data as the `plot` and `fplot` functions, just in different forms.

Specialized 2D plotting functions: `area`, `stem`, and `stairs`

- `area` plots y versus x and fills the area between 0 and y. For an example of application, type:
  ```
  x=linspace(-3*pi,3*pi,100); y=sin(x)./x;
  area(x,y)
  ```
- `stem(x,y)` plots the data sequence y at the values specified in y.
  `stem(x,y,'LineSpec')` uses the line type specified for the stems and markers. See `plot` for possibilities. For an example, type:
  ```
  x = 0:0.1:5; y = cos(x.^2).*exp(-x);
  stem(x,y,'b-')
  ```
- `stairs(x,y)` draws a stairstep graph of the elements in vector y at the locations specified in x. Example:
  ```
  x = 0:0.25:10; y=sin(x); stairs(x,y)
  ```

Specialized 2D plotting functions: scatter, bar, barh, histogram, and pie

- scatter creates a plot using circle markers. Example:
  ```
  year=2020:2023; population=[5.2 5.3 5.1 5];
  scatter(year,population)
  ```
- bar draws a bar chart. Example:
  ```
  x = -3:0.2:3;y = exp(-x.^2);bar(x,y)
  ```
- barh draws a horizontal bar chart. Example:
  ```
  x = -3:0.2:3;y = exp(-x.^2);barh(x,y)
  ```
- histogram(y,n) takes the values in the vector y and put them into n bins. Example:
  ```
  y=rand(50,1);histogram(y,10)
  ```
- pie(x,labels) specifies text labels for the slices. x must be numeric. The number of labels must equal the number of elements in x. Example:
  ```
  pie([2 4 3 5],{'Mary','John','Steve','Nick'})
  ```

Specialized 2D plotting functions: `comet` and `fimplicit`

- `comet(x,y)` displays an animated comet plot of vector `y` versus `x`. Example:

  ```
  x=linspace(0,10*pi,1000);y=x.*sin(x);
  comet(x,y)
  ```

- `fimplicit(fun,limits,'LineSpec'` plots the curves where `fun`(x,y)=0 between the axes `limits`, with a default range of [-5 5]. 'LineSpec' gives line specification. Example:

  ```
  fimplicit(@(x,y) x.^2/4+y.^2/9-1,[-4 4 ...
  -4 4],'b-')
  ```

Specialized 2D plotting functions: `polar` and `ezpolar`

- `polar(theta,r)` makes a plot using polar coordinates of the angle theta, in radians, versus the radius r.
  `polar(theta,r,S)` uses the linestyle specified in string S. See the command `plot` (above) for a description of legal linestyles. Example:
  ```
  theta=linspace(0,2*pi,201);
  r=sqrt(abs(2*sin(5*theta)));
  polar(theta,r,'g-')
  ```
- `ezpolar(fun,[a b])` plots the polar curve r=`fun` over the interval [a, b]. Example:
  ```
  ezpolar(@(t)sqrt(abs(2*sin(5*t))),[0 2*pi])
  ```

Multiple Data Sets in one Plot

Multiple (x,y) pairs arguments create multiple graphs with a single call to `plot`. For example,

```
plot(x1,y1,'r',x2, y2,'g--o')
```

plots two curves: the first is a red, solid line and the second is a a green, dashed line with circles at the data points.

We can also plot the first curve, and then add the second by using `hold on`, which is a toggle that freezes the current graph in the Figure Window, so that new graphs will be superimposed on the current one. The command `hold off` ends the process.

```
plot(x1, y1,'r')
hold on
plot(x2, y2,'g--o')
hold off
```

Note that the axes can change for every new curve. However, all the curves appear on the same plot.

You can display m plots vertically and n plots horizontally in one graphics window by

`subplot(m,n,p)`

This divides the graphics window into m-by-n rectangles and selects the p-th rectangle for the current plot. All the graphics commands work as before, but now apply only to this particular rectangle in the graphics window. You can “bounce” between these different rectangles by calling subplot repeatedly for different values of p. If you are comparing a number of plots, it is important to have the endpoints of the axes be the same in all the plots. The `sgtitle` function can be used to put a title on the entire Figure Window. An example is included in the next slide.

The following code uses `subplot` to plot the functions $\sin(x)$ and $\cos(x)$ on $[0, 2\pi]$ using 200 data points.

```
x=linspace(0,2*pi,200);
y=cos(x);
z=cos(x);
subplot(1,2,1)
plot(x,y,'b*')
grid on
xlabel('x'), ylabel('y'),title('sin(x)')
subplot(1,2,2)
plot(x,z,'r+')
grid on
xlabel('x'), ylabel('y'), title('cos(x)')
sgtitle('Graphs of sin(x) and cos(x)')
```

MATLAB has built-in functions that will display 3D plots. Many of them have the same name as the corresponding 2D plot functions with a '3' at the end.

The basic command for ploting a 3D curve is

```
plot3(xvalues,yvalues,zvalues,'style-options')
```

where 'style-options' are the same as for the 2D `plot` function.

Example:

```
t=linspace(0,10*pi,300);
x=cos(t); y=sin(t); z=t;
plot3(x,y,z,'b-*')
xlabel('x'), ylabel('y'), zlabel('z'), grid on
```

Another example:

```
t=linspace(0,10*pi,300);
plot3(exp(-0.05*t).*cos(t), ...
exp(-0.005*t).*sin(t), t, 'b-*')
xlabel('x'), ylabel('y'), zlabel('z'), grid on
```

Specialized 3D plotting functions: `fplot3`

- `fplot3(fx, fy, fz, [tmin tmax])` plots the parametric curve `fx(t), fy(t)` and `fz(t)` over the interval `[tmin, tmax]`. The default interval is `[-5, 5]` if the interval is missing.
- `fplot( ... ,'LineSpec')` plots with the given line specification. For example,

```
fplot3(@(t) sin(2*t),@(t) cos(t), ...
@(t) sin(3*t+2), [-pi,pi], 'b-')
xlabel('x'), ylabel('y'), zlabel('z')
```

  plots the curve `x=sin(2t), y=cos(t), z=sin(3t+2)` one the interval $[\pi, \pi]$ in blue with a dashed line style.

Specialized 3D plotting functions: `mesh` and `surf`

- `mesh` draws a wireframe mesh of 3D points. Example
  ```
  [x,y] = meshgrid(0:.05:2,-4:.05:4);
  z = x .* exp(-x.^2 - y.^2); mesh(z)
  ```
  Related functions: `meshc` and `meshz`
- `surf` creates a surface plot by using colors to display the the surfaces. Example:
  ```
  [x,y] = meshgrid(0:0.01:2, -4:0.01:4);
  z = x .* exp(-x.^2 - y.^2);
  surf(x,y,z)
  ```
  Related functions: `surfc`, `surfl`, `ezsurf`, `ezsurfc`, etc.

MATLAB also has several buil-in functions for specific shapes, such as `sphere` and `cylinder`

Specialized 3D plotting functions: `fimplicit3` and `contour`

- `fimplicit3` plots implicit surfaces.
  `fimplicit3(fun)` plots the surface where fun(x,y,z)=0 between the axes limits, with a default range of $[-5, 5]$.
  Example:
  ```
  fimplicit3(@(x,y,z) x.^2+y.^2+z.^2 - 9)
  ```
  `fimplicit3(fun,limits)` uses the given limits. `limits` can be [xyzmin, xyzmax] or [xmin xmax ymin ymax zmin zmax].
  `fimplicit3(...,'LineSpec')` plots with the given line specification, using the color for the surface.
- `contour` draws contour curves. Example:
  ```
  [x,y] = meshgrid(0:.1:2, -4:.1:4);
  z = x .* exp(-x.^2 - y.^2);
  cs=contour(x,y,z,[0.1 0.2 0.3 0.4 0.41]);
  clabel(cs)
  ```
  Related function: `contour3`