

GRAPHICS IN MATLAB: BASIC PLOTTING



PLOTTING A SINGLE ARRAY

- The most basic command that puts a graphic on screen is *plot*
- If *y* is an array, `plot(y)` simply plots values of *y* vs. their array index position
- Example:
 - `t=[0:0.01:1];`
 - `y=cos(2*pi*t);`
 - `plot(y)`



PLOTTING ONE ARRAY AGAINST ANOTHER

- If t is one array and y another, `plot(t,y)` plots y vs. t .
- For example,
 - `t=[0:0.01:1];`
 - `y=cos(t);`
 - `plot(t,y)`





INTERESTING EFFECTS

- Plotting one array against another generates creative figures
- For example, define your x-array as $\cos(2(\pi)t)$ and y-array as $\sin(2(\pi)t)$. Then plot y vs.x
- Do the same for $2\cos(2(\pi)t)$ vs. $4\sin(2(\pi)t)$



PLOTTING MATRICES

- The argument of *plot* can be a matrix. In this case, each column is plotted separately on the same axis. The x axis runs through index positions

- Try

- `t=0:pi/16/:2*pi;`
- `x=[cos(t)',sin(t)']`
- `plot(x)`

5	8	9	0
3	5	2	3
6	6	5	4
8	4	6	8

↑ ↑ ↑ ↑

each column plotted
separately vs. position



PLOTTING A MATRIX AGAINST AN ARRAY

- Let Y be a matrix and x be a vector, then
 - **plot(x,Y) plots rows or columns of Y vs. x**

- Try

- **t=0:pi/16:2*pi;**
- **Y=[cos(t)',sin(t)']**
- **plot(t',Y)**

5	8	9	0
3	5	2	3
6	6	5	4
8	4	6	8

↑ ↑ ↑ ↑

each column plotted separately vs. another array



PUTTING MULTIPLE PLOTS ON ONE GRAPH

- There are 3 ways to do this. Want to plot 3 data vectors y_1, y_2 and y_3 vs. t
- Method 1:
 - $Y=[y_1, y_2, y_3]$
 - `plot(t, Y)`
- Method 2
 - `plot(t, y1, t, y2, t, y3)`
- Method 3
 - Use *hold* command



hold COMMAND

- hold cycles between on and off.
- Once it is turned on, all subsequent plot commands are overwritten on the active figure window
- For example,
 - **plot(t,y1)**
 - **hold on**
 - **plot(t,y2)**
 - **plot(t,y3)**
 - **hold off**



PRACTICE

- Plot the following function over the range $x=[-2,2]$.

$$y = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$



PLOTTING PARAMETRIC FUNCTIONS- *fplot*.

- you may want to plot a function, rather than data, against a variable.
- Example is plotting $\text{sinc}(x)$. The command is

`fplot('sinc(x)',[-3 3])`



function



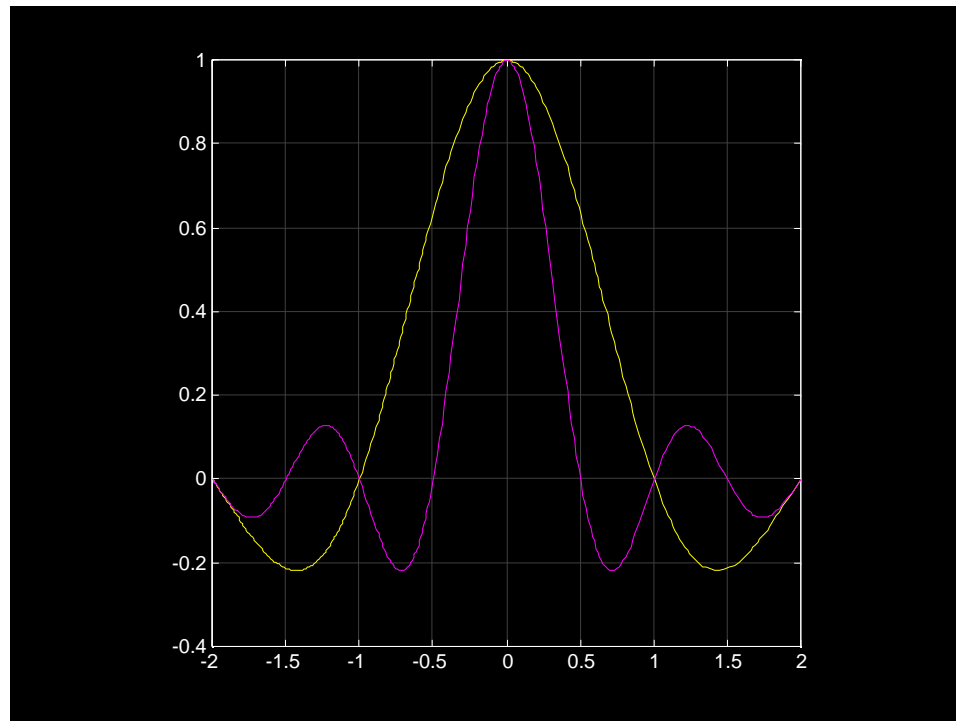
x range





MULTIPLE PLOTS USING *fplot*

```
fplot('[sinc(x),sinc(2*x)]',[-2 2])
```





EXAMPLE: RC CIRCUITS

- Capacitors charge and discharge following an exponential curve. Plot the following 3 voltages on the same graph for $0 < t < 5$.

$$v_1 = 1 - e^{-t}$$

$$v_2 = 1 - e^{-2t}$$

$$v_3 = 1 - e^{-3t}$$



CONTROLLING LINE TYPES

- We can select line styles and colors in a graph
- Simply pass a character string to plot
 - `plot(t,x,'s')`
- For example, `s=r+`, puts **red** plus(+) marks for every data point.
- MATLAB accepts many other choices.
- Redo the exponentials using different styles (use `plot` instead of `fplot`)



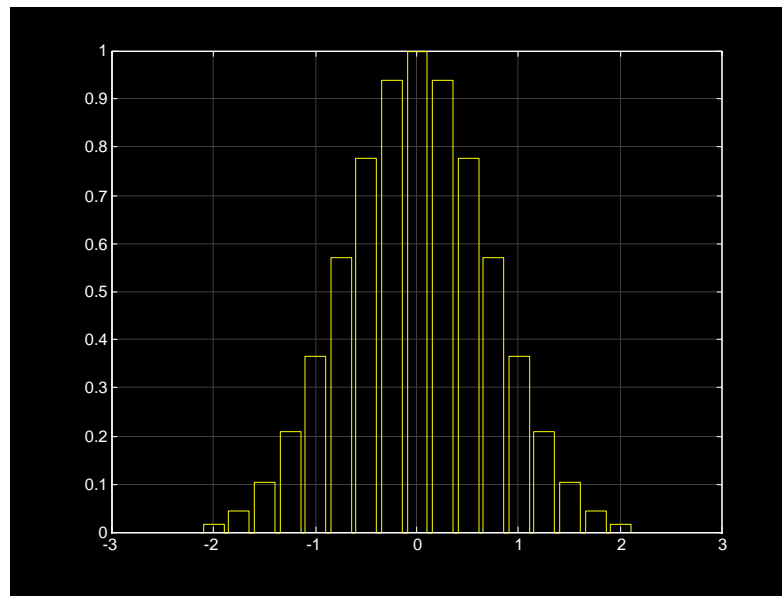
SPECIALIZED 2D PLOTTING FUNCTIONS

- *bar* - creates *polar* - creates a (,) plot
- *hist* - creates a histogram of data
- *fill* - fills the curve with solid colors
- *quiver* - creates vector plots
- *stairs* - similar to bar but without internal lines
- *rose* - creates an angle histogram

bar

- Usage:

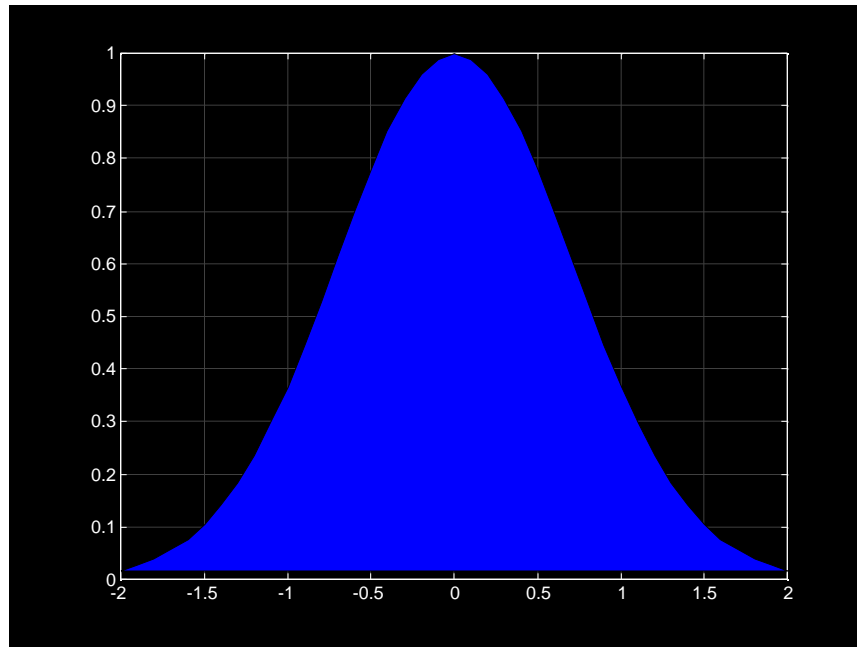
- `bar(x,y)`
- plot the shape of a normal distribution given by $y=\exp(-x^2)$ in the range $(-2,2)$.





- Usage:

- `fill(x,y,'b')` where `b` indicates a blue fill.
- plot the function $\exp(-t^2)$, in the range $(-2,2)$

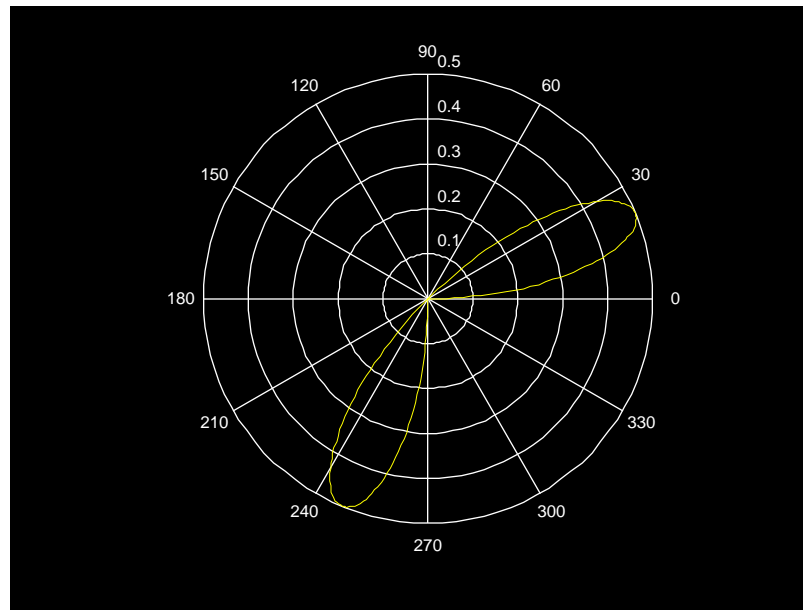


polar

- Usage:

- **polar(theta,rho)**

- Plot $= \sin(2 \) \cos(2 \)$





ADDING LABELS AND TITLES

- There are numerous ways the appearance of a plot can be controlled. The most obvious ones are through
 - axis labels
 - figure title
 - legend
 - -grid



LABEL USAGE

- **x/y labels**
 - **xlabel('time')**
 - **ylabel('voltage')**
- **Figure title**
 - **title('Capacitor voltage')**
- **Legend**
 - **gtext('RC=10⁻⁶')**
 - **places the enclosed string on the graph**
- **Grid**
 - **grid - places a grid over the plot**



3-D PLOTTING

- There are numerous ways to display a function in 3-D in black and white as well as color
- The direct extension of 2-D plot function, *plot(x,y)*, to 3-D is *plot3(x,y,z)*



PLOTTING A CORKSCREW

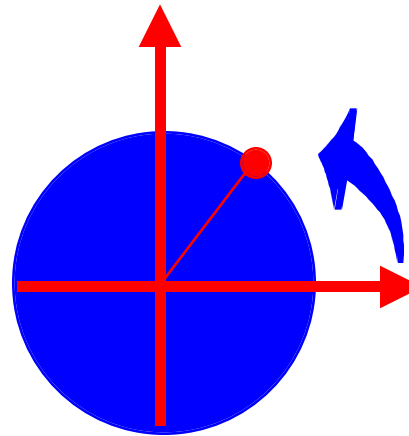
- How would you model a corkscrew?
- Corkscrew, or spiral, is the 3-D equivalent of a spiral
- It goes around a circle but it also rises from the ground plane. So, what is its equation?

3-D EQUATION

- A circle can be parametrically described by

- $x = \cos(t)$

- $y = \sin(t)$

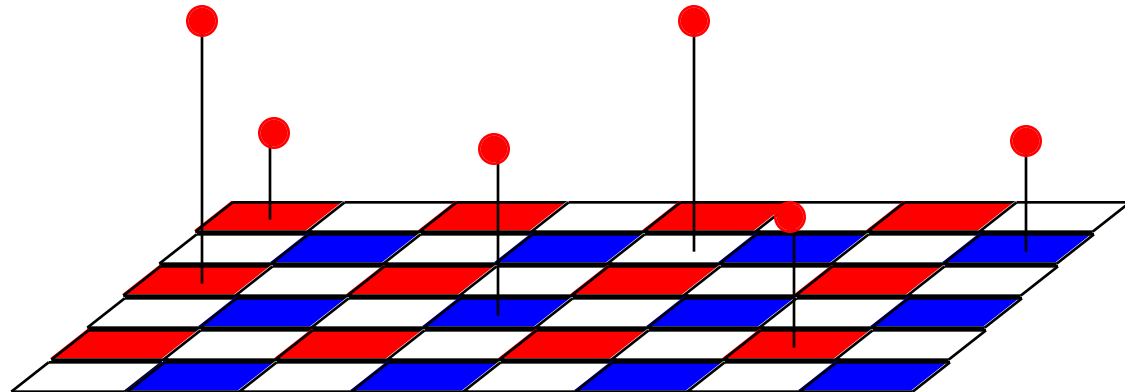


- To make it rise from the ground plane, let $z=t$ and run t from 0 to 10π .



DISPLAYING A FUNCTION AS A SET OF HEIGHTS

- A 3-D plot can be interpreted as heights above the ground plane. These heights are evaluated at some predefined grid points





mesh and meshgrid

- Key functions here are *mesh* *meshgrid*.
- Let's say we want to sample the ground plane into a grid of points;
 - $x=-8:.5:8;$
 - $y=x;$
 - $[X,Y]=\text{meshgrid}(x,y)$
- X and Y are *matrices*
- *mesh* evaluates the function over the grid

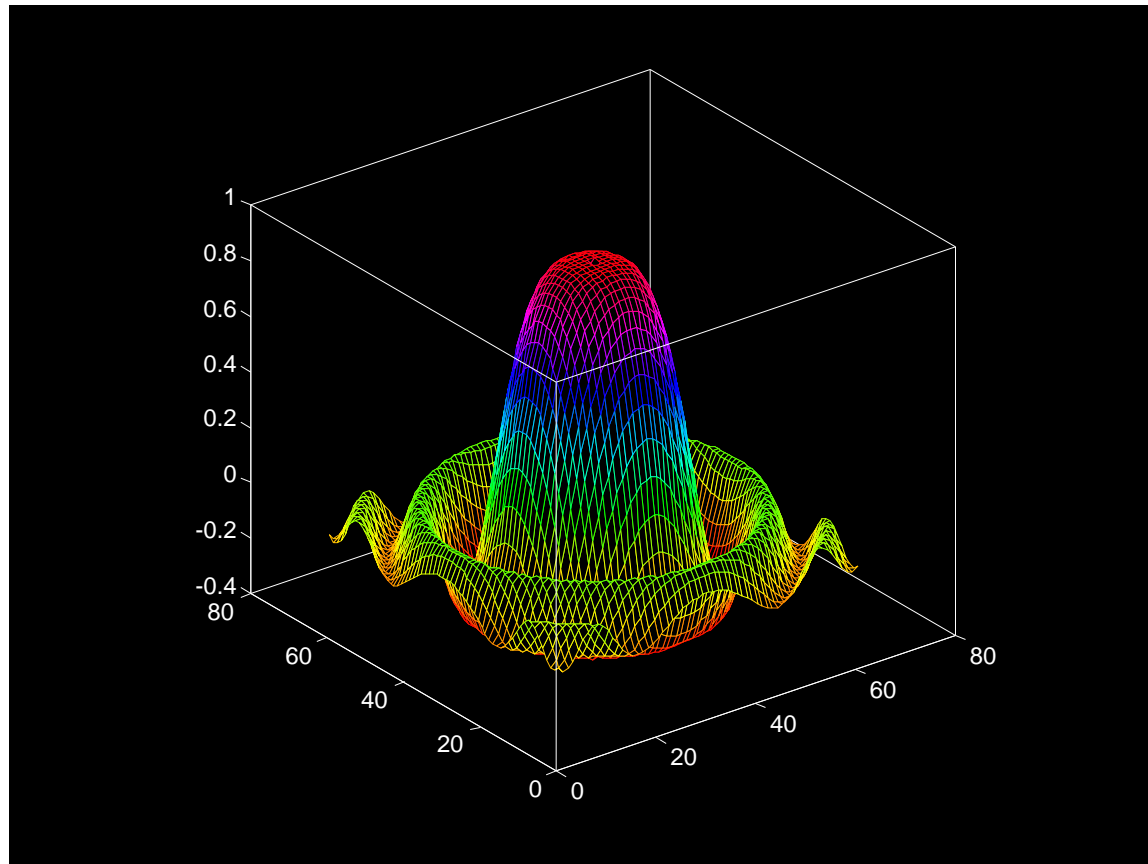


PLOTTING SOMBRERO

- Sombrero, looking like a Mexican hat, is defined by $\sin(R)/R$.
- R is the distance of a point (X,Y) to the origin, i.e.
 - $R^2 = X^2 + Y^2$
- Then,
 - $R = \sqrt{X.^2 + Y.^2}$
 - $Z = \sin(R)./R$;
 - `mesh(Z)`



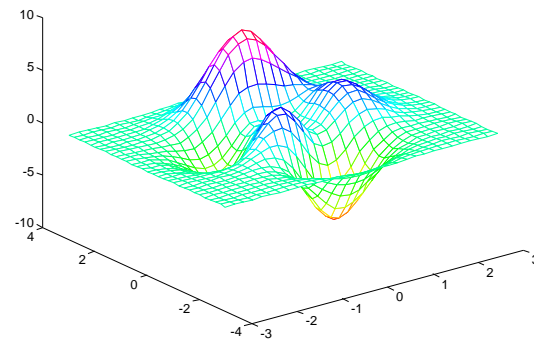
SOMBRERO





GENERATING TRUE AXIS UNITS

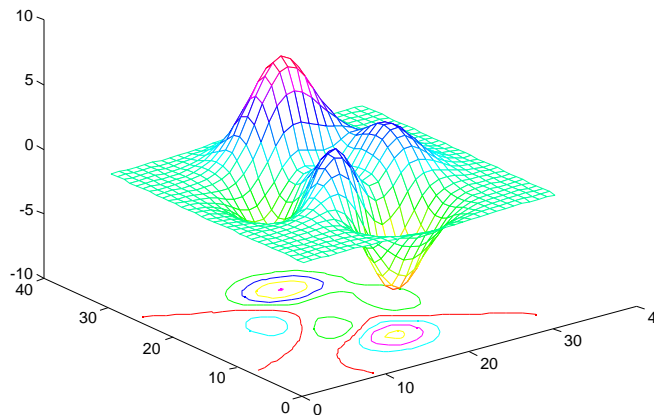
- Use of *mesh*(Z) plots Z vs. index positions
- To plot Z vs. actual units of X and Y, generate two 1D arrays x and y vector
 - `[X,Y]=meshgrid(-3:0.2:3)`
 - `Z=peaks(X,Y);`
 - `x=-3:0.2:3;`
 - `y=x;`
 - `mesh(x,y,Z)`





CONTOUR PLOT

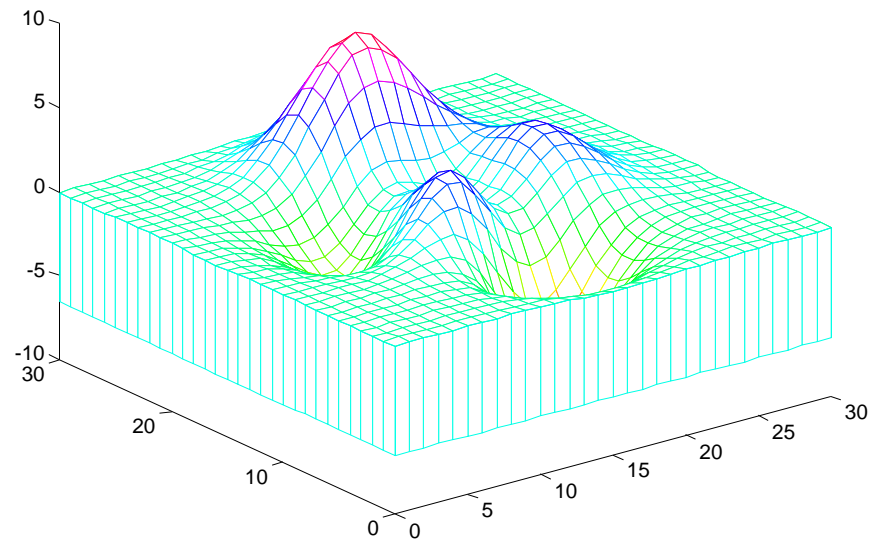
- Contours are slices of constant height that are then projected onto the ground plane
- In its simplest form *meshc* (Z) does the job





CURTAIN PLOT

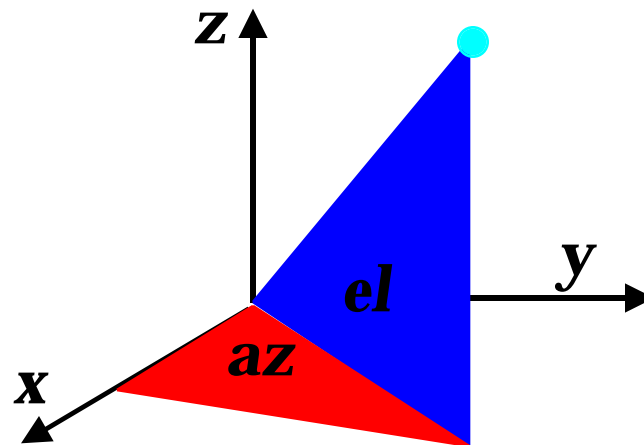
- You can put your plot on a 'pedestal' by using *meshz* (Z)





CONTROLLING VIEWPOINT

- Viewpoint is controlled by two angles: azimuth and elevation
- Azimuth is rotation around the Z-axis
- Elevation is rising above the ground plane





DEFAULT VIEWPOINTS

- In MATLAB, default viewpoints are $az=-37.5$ and $el=30$ degrees
- Zero degrees azimuth is like looking up the first column of the Z matrix.
- Zero degrees of elevation is like looking “edge on” the Z matrix



INTERPRETING SIGNS OF VIEWPOINT ANGLES

- Increasingly negative azimuth angle corresponds to holding the object in front of you and rotating it *counterclockwise*.
- Equivalently, it corresponds to keeping the object stationary and moving around it *clockwise*
- Positive elevation angle mean *rising above* the object. Elevation of +90 degrees means being directly overhead and looking down

IN-CLASS PRACTICE

- Do the following on

$$z = \sin\left(\sqrt{x^2 + y^2}\right)$$

- do a surface plot, title and label all axis
- visually find out how deep the hole is?
- what is happening inside (looking underneath)
- generate 3D contours (30 of them)
- produce level curves that are not obstructed by the object



HOMework #1

- **3D plotting and manipulations**
 - **Evaluate Sombrero and plot it for 3 interesting view points**
 - **Produce plots that show contours and curtains**
 - **Write a procedure that would cap the plot to 70% of its peak value then plot it. Your plot should show a flat top sombrero**

HOMEWORK #2

$$r_1 = 1 + \cos(\theta)$$

$$r_2 = \frac{1}{2}$$

Find out the angular position in degrees where the two curves are closests. Run theta through full circle in increments of 1 degrees

