# [QI;MP]

## Graphics

Tara Hennessy

OIST

# Outline

- 2D graphics
  - Multiple plots in one window
  - The **`hold`** command
  - **`scatter`** plots
  - **`polar`** plot
- 3D graphics
  - **`meshgrid`**
  - **`contour`**, **`contour3`**
  - **`mesh`** and **`surf`** and **`pcolor`**
  - Printing and saving
  - Titles and labels
  - **`subplot`** and **`view`**point
- Other features of *Matlab*
  - **`lighting`**
  - **`ginput`**
  - **`colormapeditor`**

# 2-D Graphics



For 2-D graphics the basic command is:

**`plot(x1, y1, 'line style', x2, y2, 'line style'....)`**

This command plots vector `x1` against vector `y1`, vector `x2` against vector `y2` etc. on the same graph.

Also; **`polar, bar, stairs, scatter`**

**Example 1.** Plot $y_1=sin(x)$ and $y_2=cos(x)$ with $x$ in $[0, 2\pi]$ on the same graph. Use a solid line for $sin(x)$ and the symbol + for $cos(x)$.

The first step is to define a set of values for x at which the functions will be defined.

# 2-D Graphics

[QI;MP]

For 2-D graphics the basic command is:

**`plot(x1, y1, 'line style', x2, y2, 'line style'....)`**

This command plots vector `x1` against vector `y1`, vector `x2` against vector `y2` etc. on the same graph.

Also; **`polar, bar, stairs, scatter`**

**Example 1.** Plot $y_1=sin(x)$ and $y_2=cos(x)$ with $x$ in $[0, 2\pi]$ on the same graph. Use a solid line for $sin(x)$ and the symbol + for $cos(x)$.

The first step is to define a set of values for x at which the functions will be defined.

```
x=0:0.1:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'-',x,y2,'+')
```

# The **hold** Command

Another way to get multiple plots on the same graph is to use the hold command to keep the current graph, while adding new plots.

Another hold command releases the previous one. For example, the following statements generate the same graph as in **Example 1.**

**Example 2.**

```
clf

x=0:0.1:2*pi;
plot(x, sin(x),'-')

hold on

plot(x,cos(x),'+')

hold off
```

# `scatter` plots



**`scatter(X,Y)`** displays circles at the locations specified by the vectors X and Y. This type of graph is also known as a bubble plot.

**Example 3.** Set up a vector x. Set up a vector y to contain cosine values with random noise. Create a scatter plot using the two vector inputs.

# `scatter` plots



**`scatter(X,Y)`** displays circles at the locations specified by the vectors X and Y. This type of graph is also known as a bubble plot.

**Example 3.** Set up a vector x. Set up a vector y to contain cosine values with random noise. Create a scatter plot using the two vector inputs.

```
x=0:0.01:3.*pi;
y = cos(x)+ rand(1,length(x));
scatter(x,y,'+','r')
```

# 3-D Graphics



For 3-D graphics the most commonly used commands are:

```
plot3(x1, y1, z1,‘line style’, x2, y2, z2, ‘line style’....
contour(x,y,Z), mesh(x,y,Z), surf(x,y,Z)
pcolor, image, contour3
```

# 3-D Graphics



For 3-D graphics the most commonly used commands are:

**plot3(x1, y1, z1,'line style', x2, y2, z2, 'line style'....**

**contour(x,y,Z), mesh(x,y,Z), surf(x,y,Z)**

**pcolor, image, contour3**

**Example 4.**

```
x=-2*pi:4.*pi./200:2.*pi;
y=0:4.*pi./200:4.*pi;
[X,Y] = meshgrid(x,y);

Z = sin(X)+cos(Y);
contour(X,Y,Z)

%contour3(X,Y,Z,30)
%pcolor(Z)
%colorbar
```

# Printing and Saving



You can save matrices into .mat files directly from your workspace.
You can then load these matrices back in whenever you want.

```
save workspace.mat

save SpecificStuff.mat X Y Z

load filename.extension

print -djpeg100 imagename.jpg
```

# Axis commands

```
axis([xmin xmax ymin ymax zmin zmax])

axis auto
axis square
axis on
axis off
caxis([zmin zmax])
```

# **surf** and **mesh** and **pcolor**



**surf** and **mesh** are quite similiar. **mesh** plots a coloured mesh, while **surf** plots a black mesh and fills in the spaces between in colour.

**Example 5.**

```
[x,y] = meshgrid([-2:.2:2]);

       Z = x.*exp(-x.^2-y.^2);

surf(x,y,Z)
colorbar
colormap jet

%shading interp
%mesh(x,y,Z)
%pcolor(Z)
%gradient(Z)
```

# Polar Coordinates



**Example 6**. Plot $\varrho=\theta^2$ with $0\leq\theta\leq 5\pi$ in polar coordinates.

```
theta=0:0.2:5*pi;
rho=theta.^2;
polar(theta,rho,'*')
```

**Exercise**. Plot $\varrho=sin(2\theta)cos(2\theta)$ with $0\leq\theta\leq 2\pi$ in polar coordinates.

# Another Example



**Example 7.** Plot $z = sin(r)/r$ with $r = \sqrt{x^2 + y^2}$, $-8 \le x \le 8$, $-8 \le y \le 8$.

The first step in displaying a function of two variables, $z = f(x,y)$, is to use the meshgrid function to generate X and Y matrices consisting of repeated rows and columns, respectively, over the domain of the function.

The function can then be evaluated and graphed.

# Another Example



**Example 7.** Plot $z=sin(r)/r$ with $r=\sqrt{x^2+y^2}$, $-8 \le x \le 8$, $-8 \le y \le 8$.

The first step in displaying a function of two variables, $z=f(x,y)$, is to use the meshgrid function to generate X and Y matrices consisting of repeated rows and columns, respectively, over the domain of the function.

The function can then be evaluated and graphed.

```
x=-8:0.5:8;
y=x;
[X,Y]=meshgrid(x,y);
R=sqrt(X.^2+Y.^2)+eps; %add eps to prevent R=0;
Z=sin(R)./R;
mesh(x,y,Z)
```

# Titles and Labels

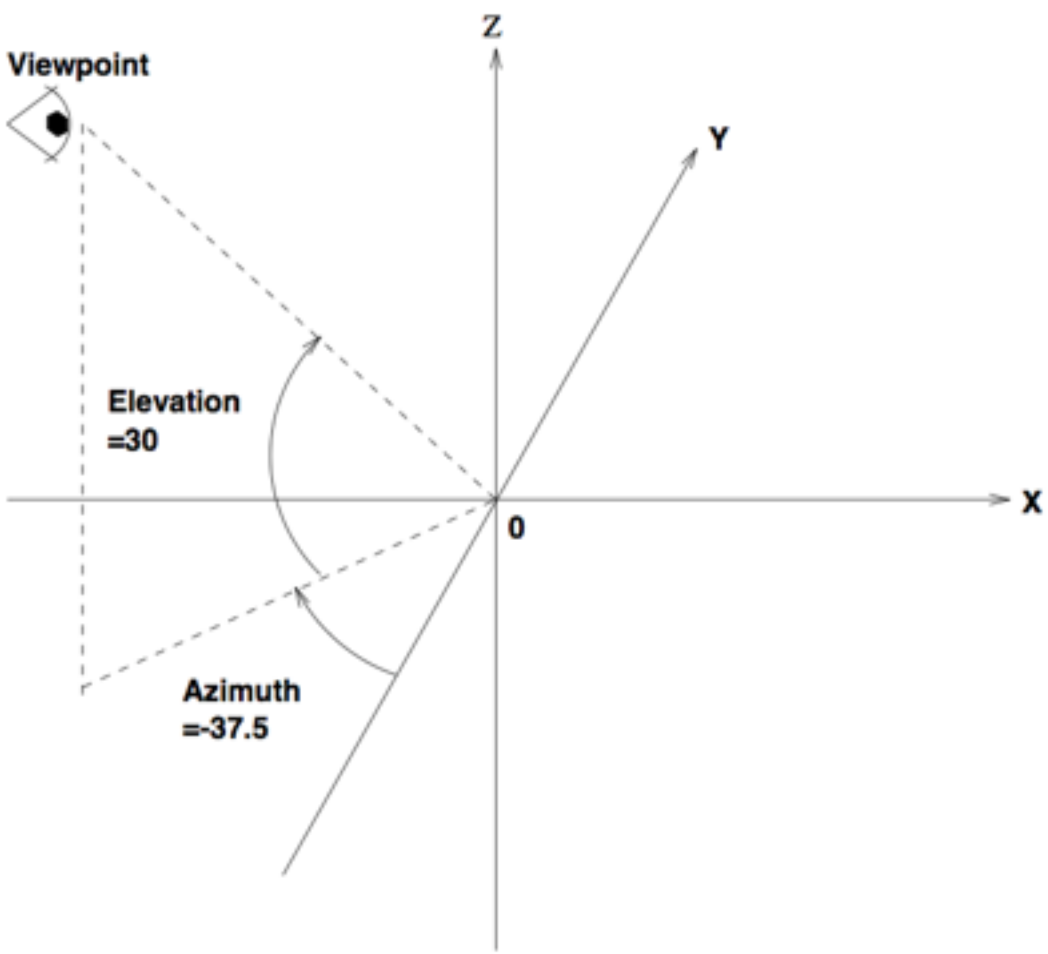
**Example 8.** Plot y=sin(x) with $0 \leq x \leq 2\pi$ with appropriate labels.

```
x=0:0.1:2*pi;
plot(x,sin(x))

     title('Y=sin(X)')
     xlabel('X');
     ylabel('Y');
hold
plot(pi,0,'*')
text(pi+0.1, 0,'Critical Point')


%gtext('critical point')
```

# `subplot` and `view`

The command **subplot(m,n,p)** breaks the graph window into an m-by-n matrix of small rectangular panes. The value of **p** is the pane for the next plot. To return to the default single figure per window use **subplot(1,1,1)** or **clf**.

You can have more than one graphics window on a X display. The *Matlab* command **figure** opens a new window, numbering each new window.




Default Viewpoint

**Example 9.** Display the internal *Matlab* **peaks** matrix from 4 different viewpoints.

```
subplot(2,2,1); mesh(peaks(20)); view(-37.5,30)
subplot(2,2,2); mesh(peaks(20)); view(-7,80)
subplot(2,2,3); mesh(peaks(20)); view(-90,0)
subplot(2,2,4); mesh(peaks(20)); view(-7,-10)
```

# Lighting



**Phong** lighting is good for curved, interpolated surfaces. **gouraud** is also good for curved surfaces

## Example 10.

```
points=0:0.001:2;
[X, Y] = meshgrid(-points, points);

Z = 2./exp((X-.5).^2+Y.^2)-2./exp((X+.5).^2+Y.^2);

surf(X, Y, Z);
shading interp;

lightangle(75, 10);
lighting phong;
view(30, 30);
```

# Lighting



**Example 11.**

```
x=-4.*pi:0.01:4.*pi;
y=0:0.01:8*pi;
[X,Y] = meshgrid(x,y);

     Z = 2.*sin(X).^2+cos(Y).^2+2.*exp((X+Y)./30);

mesh(X,Y,Z)
shading interp
axis off
lightangle(35,50)
view(-10,55)

%lighting gouraud
```

# ginput



Graphical input from mouse or cursor.

**Example 12.**

Pick 8 two-dimensional points from the figure window. Draw a zig zag!

```
[x,y] = ginput(8)
```

Position the cursor with the mouse. Enter data points by pressing a mouse button or a key on the keyboard. To terminate input before entering 4 points, press the **Return** key.

# Make this one pretty!



**Exercise.**

```
load ScatteredCylindricalWave.mat
P=ScatteredFieldWithoutFibre;
mesh(P);
```