JOINT INSTITUTE
交大密西根学院

VG101: Introduction to Computers and Programming

Lecture 05

Xiaodong Wei

Outline

- Control statements cont.
- Random Numbers and Monte Carlo Method
- 2D Plot
- 3D Plot

Practice

- Given n, create a matrix as follows

1, 2, 3, …, n-2, n-1, n
2, 3, 4, …, n-1, n, 1
3, 4, 5, …, n, 1, 2
…
n, 1, 2, …, n-3, n-2, n-1

Continue and Break in Loops

- The `continue` statement passes control to the next iteration of the **for** or **while** loop in which it appears, skipping any remaining statements in the body of the loop.
- The `break` statement terminates the execution of a **for** loop or **while** loop. When a break statement is encountered, execution continues with the next statement outside of the loop.

Example

```
clear all; clc;
n = 1;
while true
    fprintf('n=%d\n', n);
    str = input('User input: ', 's');
    switch str
        case 'a'
            n = n+1;
        case 'b'
            continue;
        otherwise
            break;
    end
end
```

JOINT INSTITUTE
交大密西根学院

Random Numbers and Monte Carlo Method

JOINT INSTITUTE
交大密西根学院

Generating Pseudorandom Numbers

- Pseudorandom numbers: appear to be statistically random, despite having been produced by a completely deterministic and repeatable process.
- The process is a computer algorithm called a *pseudorandom number generator*, which takes an input number called a *random seed*: same seed, same "random" number.
- `rand`: uniformly distributed random numbers between [0 1].
- `randn`: normally distributed random numbers. The normal distribution has mean 0 and standard deviation 1.

Change the rand State

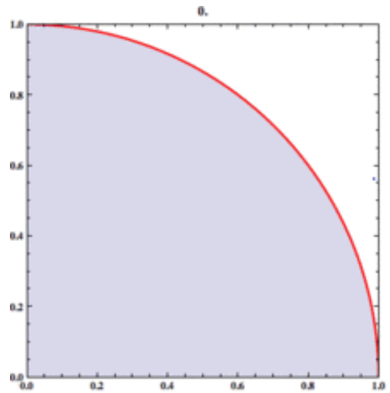
- MATLAB resets the `rand` state at startup. `rand` generates the same sequence of numbers unless you change the value of the state input.
  - Example: `rand(2,2)` % restart MATLAB
- Tip: you can use the current date and time as the `rand` state, so that every time you run the program, the `rand` state will be different.
  - `rng('shuffle')`

Monte Carlo Method

- Rely on repeated random sampling to compute the results.
- In contrast to deterministic algorithms, e.g., numerical integration, etc.

Example: Approximate π by Monte Carlo

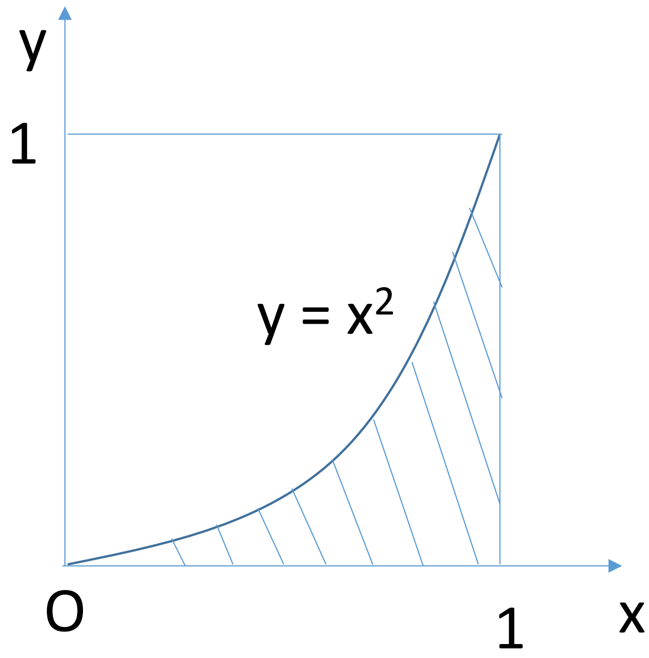
- Given that a circle inscribed in a square and the square itself have a ratio of areas that is π/4, the value of π can be approximated using a Monte Carlo method:
  - Draw a square, then inscribe a circle within it.
  - Uniformly sample points within the square.
  - Count the number of points inside the circle and the total number of points.
  - The ratio of the two counts is an estimate of the ratio of the two areas, which is π/4. Multiply the result by 4 to estimate π.



http://en.wikipedia.org/wiki/Monte_Carlo_method

Practice

- Calculate the area under the function $y=x^2$ for $0<x<1$.



$$\int_0^1 x^2 dx = \frac{1}{3}x^3\Big|_0^1 = \frac{1}{3}$$

2D Plot

Overview

- MATLAB provides a wide variety of techniques to display data graphically.
- Interactive tools enable you to manipulate graphs to achieve results that reveal the most information about your data.
- You can also annotate and print graphs for presentations, or export graphs to standard graphics formats for presentation in web browsers or other media.

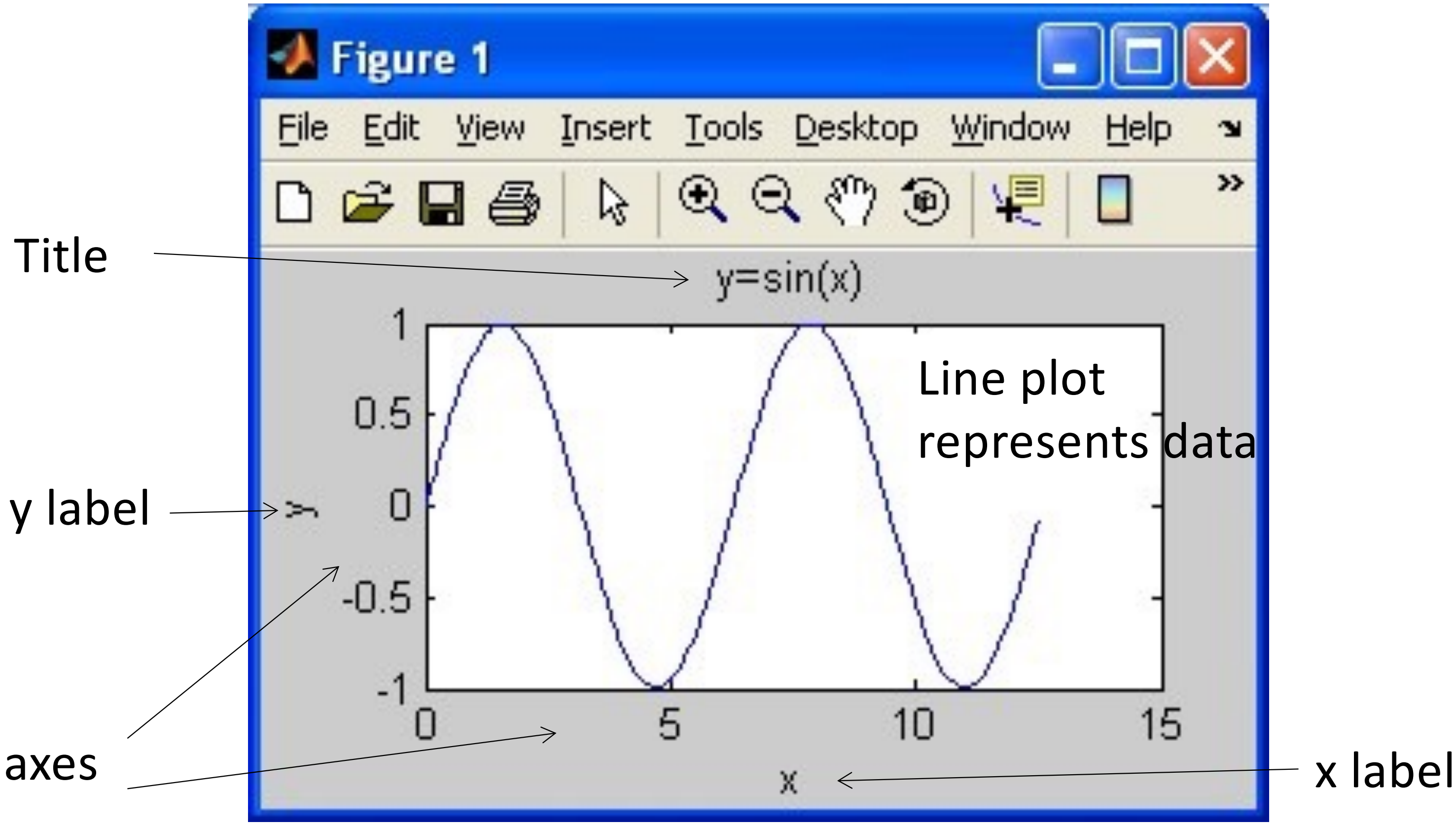
The Plotting Process

- Creating a graph: using plotting tools or functions
- Exploring data: extract specific information about the data or perform data fitting
- Editing the graph components: axes, line…
- Annotating graphs: text, arrows, labels…
- Printing or exporting graphs
- Saving graphs to reload into MATLAB

Graph Components

- MATLAB displays graphs in a special window known as a figure. To create a graph, you need to define a coordinate system. Therefore every graph is placed within axes, which are contained in the figure.
- `figure`: create a new figure window.

Example



Title
y label
axes
x label
Figure 1
File Edit View Insert Tools Desktop Window Help
y=sin(x)
Line plot
represents data
1
0.5
0
-0.5
-1
0
5
10
15
x
y

The plot Function: Linear 2D plot

Some useful variations:

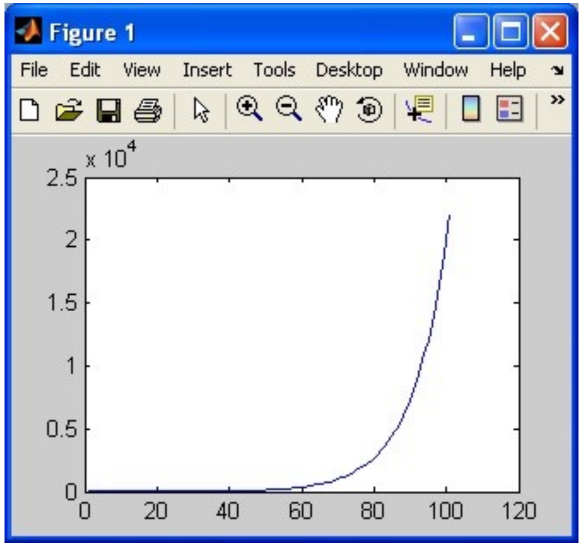
- `plot`(Y): plot Y against its indices
  - If Y is a vector, indices (i.e., x coordinates) range from 1 to `length`(Y).
  - If Y is a matrix, indices from 1 to the number of rows in Y; each column of Y yields one plot; all the plots in the same figure.
  - If Y is complex, `plot`(Y) is equivalent to `plot`(`real`(Y),`imag`(Y)), where `real`(Y) and `imag`(Y) return the real part and imaginary part of Y

- `plot`(X, Y): plots Y vs. X
- `plot`(X, Y, LineSpec): plot the line with line specifications
- `plot`(…, 'PropertyName', PropertyValue, …): plot while setting properties

JOINT INSTITUTE
交大密西根学院

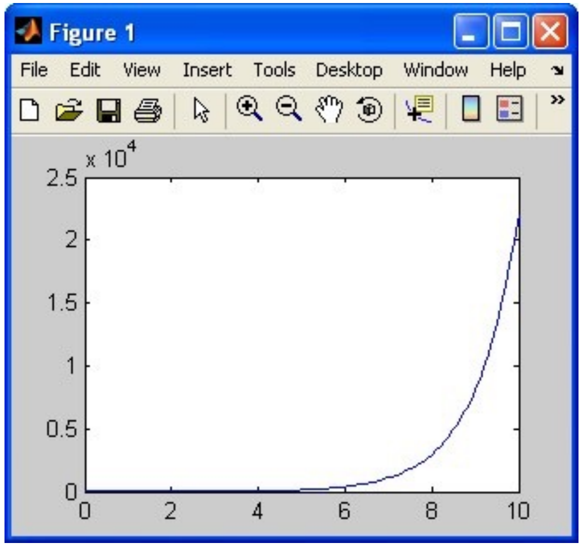
Example: plot(Y)

```
Y = exp(0:0.1:10);
plot(Y);
```



Example: plot(X, Y)

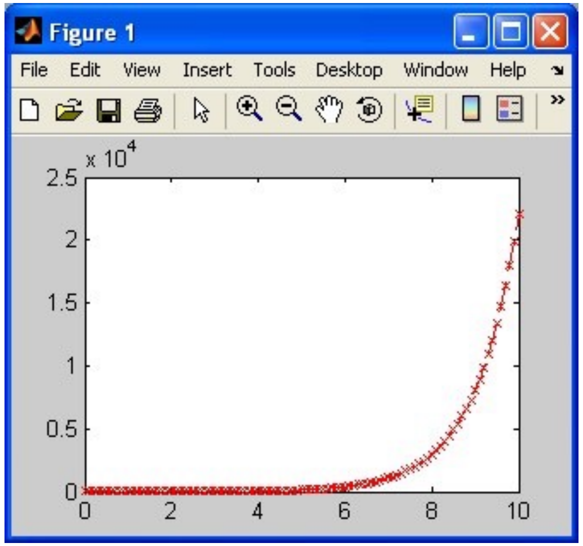
```
X = 0:0.1:10;
Y = exp(X);
plot(X,Y);
```



Example: plot(X, Y, LineSpec)

- LineSpec: line style + marker symbol + color
  - Order doesn't matter

```
X = 0:0.1:10;
Y = exp(X);
plot(X,Y,'--xr');
```



LineSpec

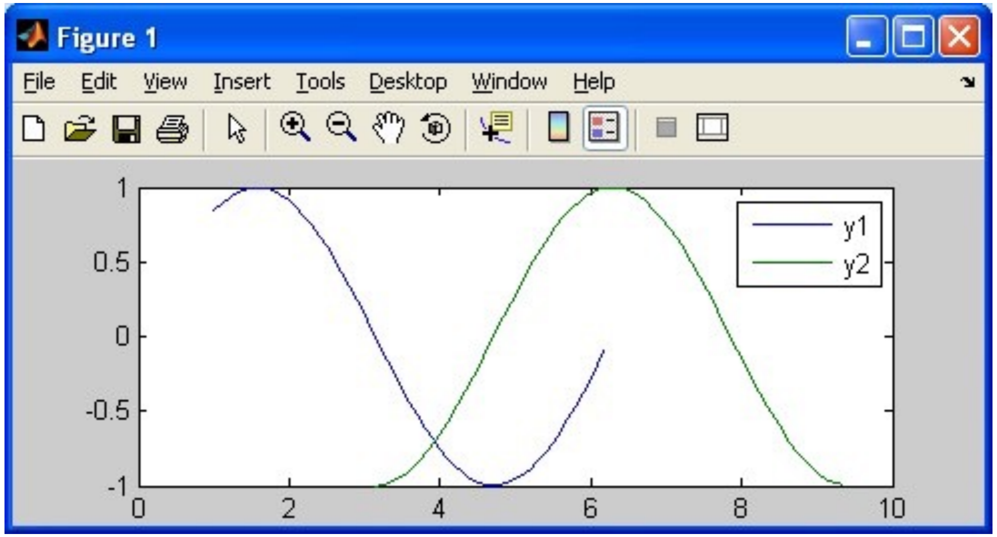
color	marker style		line style
y = yellow	. = point	o = circle	- = solid
m = magenta	x = x-mark	+ = plus	: = dotted
c = cyan	s = square	d = diamond	-. = dash-dot
r = red	* = star	p = pentagram	-- = dashed
g = green	h = hexagram		
b = blue	v = triangle down		
w = white	^ = triangle up		
k = black	> = triangle right		
	< = triangle left		

Multiple Data Sets in One Graph

- Multiple x-y pair arguments create multiple graphs with a single call to plot.
- MATLAB automatically cycles through a predefined (but user settable) list of colors to distinguish different sets of data.
- `plot`(X1,Y1,X2,Y2...)
- `plot`(X1,Y1,LineSpec1,X2,Y2,LineSpec2...)

Example

```
x1 = 1:0.1:2*pi;
y1 = sin(x1);
x2 = pi:0.1:3*pi;
y2 = cos(x2);
plot(x1,y1,x2,y2);
legend('y1','y2');
```



Practice

- Plot an equilateral triangle with Matlab

Controlling the axes

- The axis command provides a number of options for setting the limits and aspect ratio of graphs. You can also set these options interactively.
- `axis`([xmin xmax ymin ymax]): set the limits for the x- and y-axes on the current plot.
- `axis equal`: use equal data units along each coordinate direction.

Axis Labels, Titles, Grid

- `xlabel, ylabel`
- `title`
- `grid`

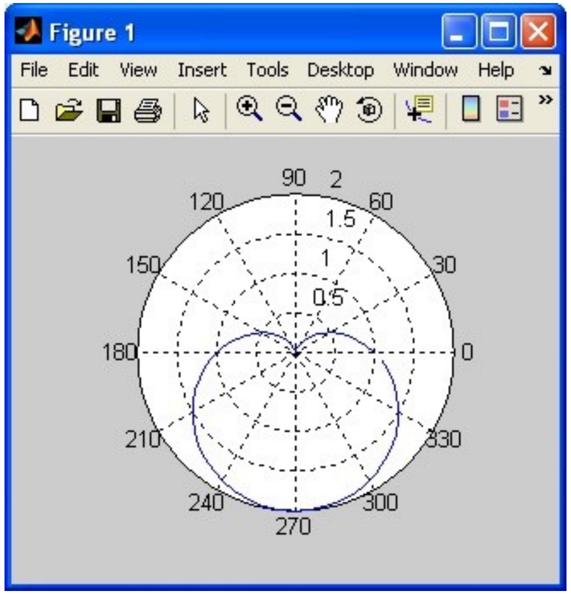
Other Two-Dimensional Plots

- `polar`(theta, r): a polar plot
- `bar`(x, y): a vertical bar chart
- `barh`(x, y): a horizontal bar chart (x = labels, y = values)
- `pie`(x): a pie chart

Example: the Heart Curve

- $r = 1 - \sin(\theta)$

```
theta = 0:0.1:2*pi;
r = 1 - sin(theta);
polar(theta,r);
```

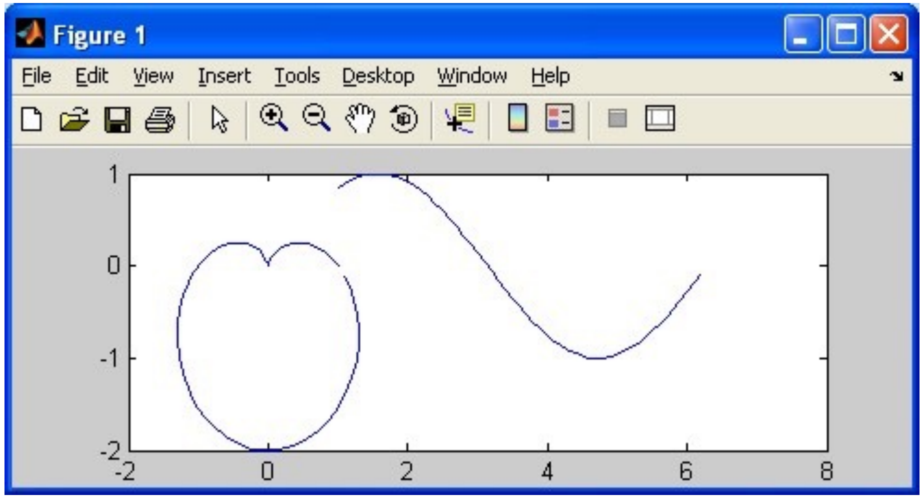


Multiple Plots in the Same Figure

- `hold on`: will prevent the graph from being redrawn and all subsequent plots will be plotted on top of old graphs.
- `hold off`: will turn off this feature
- `hold`: toggle between the two states

Example

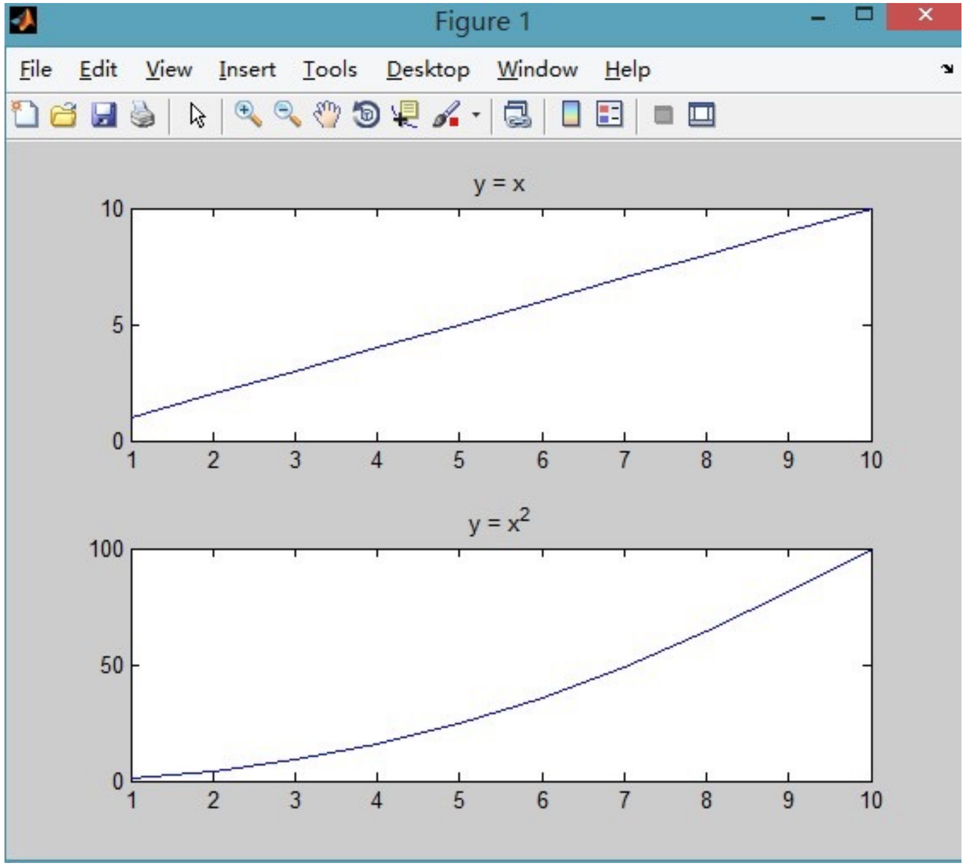
```
x = 1:0.1:2*pi;
y = sin(x);
theta = 0:0.1:2*pi;
r = 1 - sin(theta);
plot(x,y);
hold on
polar(theta,r);
```



Subplot

- `subplot`(m,n,p)
  - Divides the current figure into an m-by-n grid and creates axes in the position specified by p

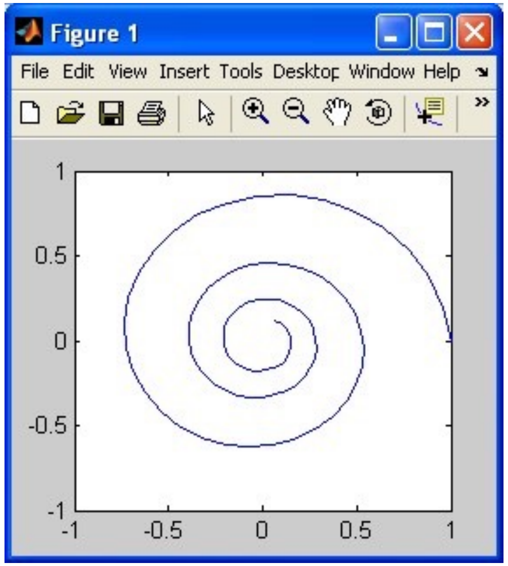
```
subplot(2,1,1);   % 2×1 subplots, the 1st one
x=0:0.1:10;
y=x;
plot(x,y);
title('y=x');

subplot(2,1,2);   % 2×1 subplots, the 2nd one
y1=x.^2;
plot(x,y1);
title('y=x^2');
```



3D Plot

The Problem of 2D Plot

```
t = 0:0.1:10;
x = exp(-0.2*t) .* cos(2*t);
y = exp(-0.2*t) .* sin(2*t);
plot(x,y);
```

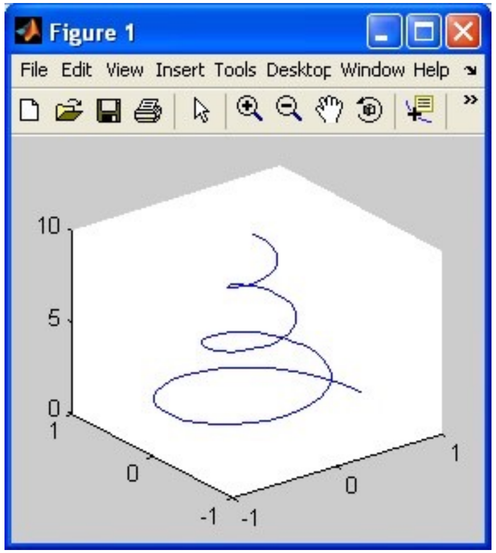


- This produces a 2D plot, but it tells us nothing about time.

plot3: 3D Line Plot

- `plot3`(x,y,z): plot coordinates in 3D space
  - x, y, z: vectors of the same length

```
t = 0:0.1:10;
x = exp(-0.2*t) .* cos(2*t);
y = exp(-0.2*t) .* sin(2*t);
plot3(x,y,t);
```



- Now we can immediately see the time dependence.

Syntax of plot3

- Similar as `plot`
- `plot3`(X1,Y1,Z1,...)
- `plot3`(X1,Y1,Z1,LineSpec,...)
- `plot3`(...,'PropertyName',PropertyValue,...)
- `h = plot3`(...)

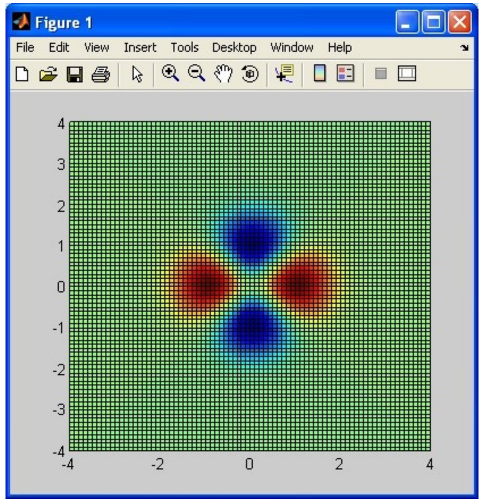
Plot of Three Dimensional Data

- 3D plots are particularly useful when we have data or functions that vary in more than one spatial dimension.
- For example let's say we wanted to know what the function

$$z = (x^2 - y^2)\ \exp(-(x^2 + y^2))$$

looks like.

Set Up a Mesh



- Use the built-in function:
  [X,Y]= `meshgrid`(x, y) % x=xstart:xincrement:xend
  - x, y: two vectors that specify the grid coordinates
  - X: a matrix where each row is a copy of x
  - Y: a matrix where each column is a copy of y
  - X and Y have the same dimension: `length`(y)-by-`length`(x)
  - `meshgrid`(x) is equivalent to `meshgrid`(x,x)
- Once the grid is set up, then computing our function is simple:

```
[x,y] = meshgrid(-4:0.1:4);
z = (x.^2-y.^2) .* exp(-(x.^2+y.^2));
pcolor(x,y,z);
```

JOINT INSTITUTE
交大密西根学院

Some 3D Plot Functions

- Pseudocolor plot: displays matrix data (z) as a grid of colored faces

```
pcolor(x,y,z);
```

- Contour plot: display automatically selected isolines of matrix z

```
contour(x,y,z);
contourf(x,y,z); % filled contour
```

- Perspective views can be obtained with

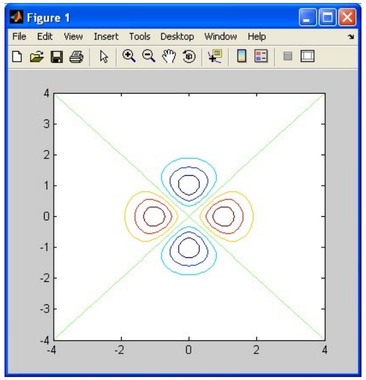
```
mesh(x, y, z); % mesh plot (only edge, no face); z defines surface heights
meshc(x, y, z); % with contour
surf(x, y, z); % surface plot (both edge and face)
surfc(x, y, z); % with contour
surfl(x, y, z); % with lighting
```

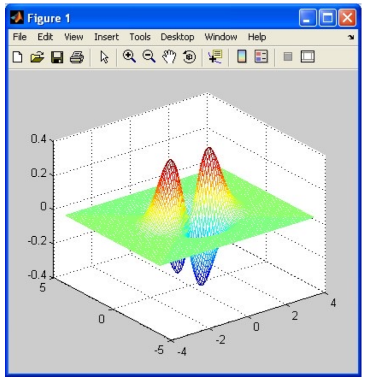
Examples

```
[x,y] = meshgrid(-4:0.1:4);
z = (x.^2-y.^2) .* exp(-(x.^2+y.^2));
pcolor(x,y,z);
```

or

```
contour(x,y,z);
mesh(x,y,z);
surf(x,y,z);
```





Options with 3D Plots

- In shaded plots (e.g., `pcolor`, `surf`) you can choose different shading options:

```
shading faceted % default, show black mesh lines
shading flat % no mesh lines
shading interp % varies the color in each face: smooth transition
```

- You can also choose different colormaps

```
colormap pink
colormap copper
```

- See the online help for more options