# Using Matlab

Jake Blanchard
University of Wisconsin - Madison
Spring 2008

# About Matlab

- Origins are in linear algebra
- Much functionality added later
- Runs on all platforms
- Many toolboxes exist

# Matlab Demos

- Type “demo”
- Poke around…

# Matlab GUIs

- Matlab has a tool for creating Graphical User Interfaces
- You can start it up by typing `guide` at the command prompt
- Let me know if you would like to know how to do these. I've got a video that describes a simple example.

# Matlab Toolkits

- Simulink: dynamic systems simulator
- Stateflow: event-driven systems
- DSP, Signal Processing, Image Processing
- Control, Optimization
- PDE, Financial, Mapping
- and many more
- We'll discuss more of this later

# Starting Out with Matlab

- Start Matlab - you'll see the "Command Window"
- Type the following:

```
A=[2 4; 1 3]
B=inv(A)
A*B
```

# Plotting

- Make vectors for x and y axis and then plot them

```
x=0:0.1:10
y=sin(x)
plot(x,y)
```

# The User Interface

- You can use Matlab interactively
- Just type commands and view results
- Difficulty is saving session
- I prefer to use scripts (m-files)
- I use the built-in editor

# My Approach

- Put commands into m-file
- Run from main Matlab window
- Edit m-file
- Rerun
- Repeat to perfection
- Save and turn in m-file

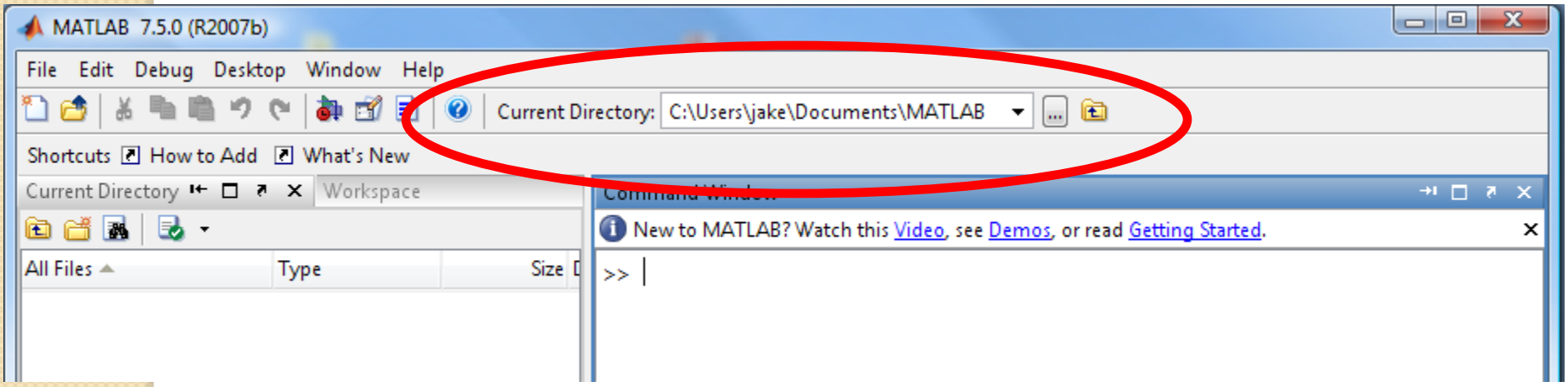
# Demo

# Key Interface Elements

- Command Window
- Current Directory
- Workspace
- Command History
- Editor
- Save to m-file from history window
- File/Save As…

# Matlab Path

- When you run a script, Matlab looks in the Matlab path for the file
- It assumes a .m extension
- Path is at top of command window



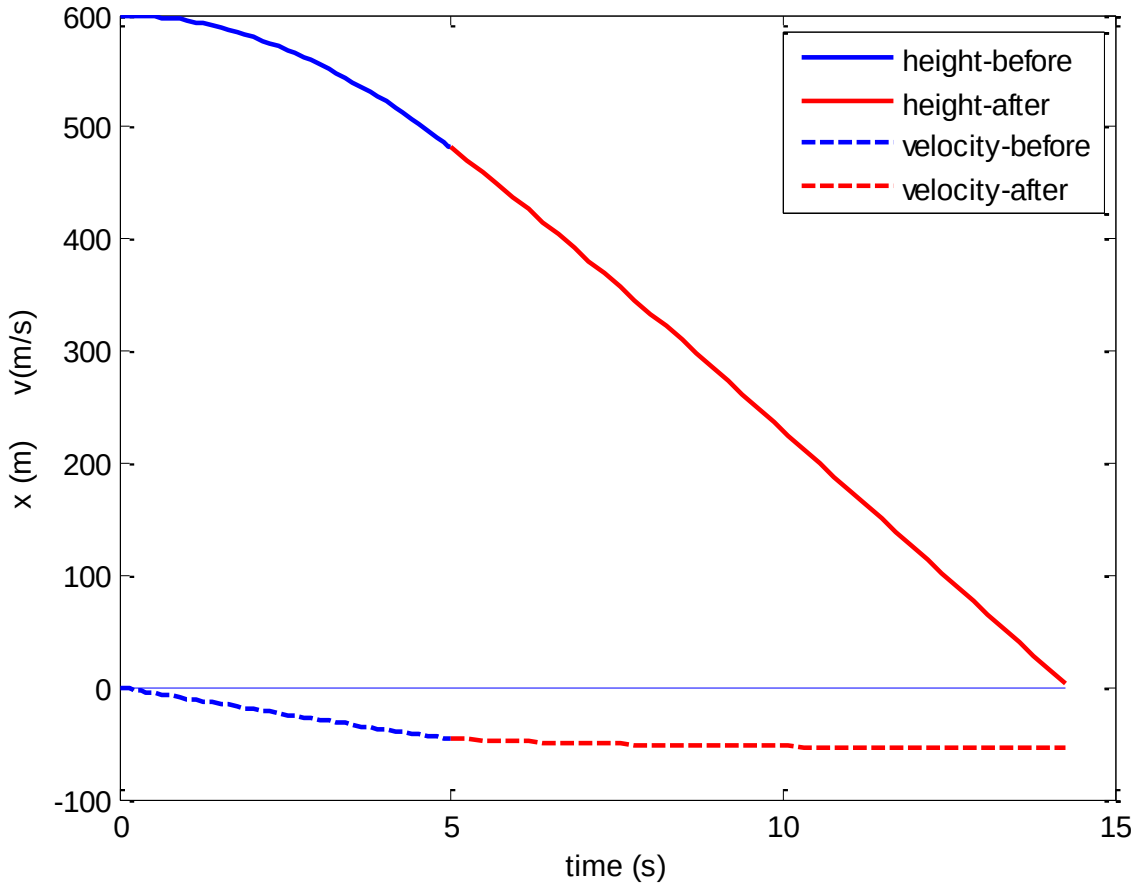
# Practice with m-files

- Download the file ***falling.m*** from course web site
- Put them somewhere in your path
- Type **falling** in command window

# The problem

- These scripts are calculating elevation for 80 kg paratrooper falling from 600 meters.
- Chute opens after 5 second free-fall.
- Drag coefficient increases by factor of 4 with chute open.
- Chute opens at ~480 meters and trooper reaches ground at about 14 s.

# Output


600
500
400
300
200
100
0
-100
0
5
10
15
x (m) v(m/s)
time (s)
height-before
height-after
velocity-before
velocity-after

# Calling Script

```
mass=80;
height=600;
gravity=9.81;
tsplit=5;
alpha=1/15;
trange=[0 tsplit];
inits=[height,0];
[t,y]=ode45(@fallfunc,trange,inits);
alpha=4/15;
trange=[tsplit 2.85*tsplit];
inits=[y(end,1) y(end,2)];
[t2,y2]=ode45(@fallfunc,trange,inits);
```

# Question

- What is impact velocity?
- What is impact velocity if chute doesn't open? [set time before chute opens (tsplit) to longer time and read off velocity when height=0]

# Variables

**A=5**

**B=3**

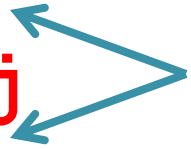
**C=A+B**

**C=C+3**

- Up to 63 characters
- Must begin with letter
- Contain letters, digits, and underscore
- No punctuation
- Case-sensitive
- No spaces

# Pre-Defined Constants

- `pi`
- `i`
- `j`



$$\sqrt{-1}$$

# Managing Variables and Screen

- `clear` – removes variables from memory
- `clear var1 var2` – removes specific variables from memory
- `clc` – clear screen
- `who` – lists currently defined variables
- up arrow will recall commands and TAB will do name completion

# Formatting output

| | |
|---|---|
| **format short** | 3.1416 |
| **format long** | 3.141592653589793 |
| **format short E** | 3.1416e+000 |
| **format hex** | 400921fb54442d18 |
| **format bank** | 3.14 |
| **format rat** | 355/113 |

# More output options

- **disp** – displays value of variable without variable name

```
>> vec=0:5;
>> disp(vec)
     0     1     2     3     4     5
>> disp(pi)
    3.1416
>> disp('This is "pi" in short format')
This is "pi" in short format
```

# More Output Options - fprintf

```
fprintf('The number pi is %f\n', pi)
The number pi is 3.141593
fprintf('The number pi is %6.2f\n‘, pi)
The number pi is   3.14
fprintf('The number pi is %6.2e\n', pi)
The number pi is 3.14e+000
```

# Getting help

- Go to help menu or type

**help plot**

# Practice

- Write a script to calculate the pressure of 1 mol of chlorine gas in a 22.4 liter container at 273 K
- Ideal gas law

$$P = \frac{nRT}{V} \qquad R = 0.08206$$

- van der Waals gas law

$$P = \frac{nRT}{V - nb} - \frac{an^2}{V^2} \qquad a = 6.49 \qquad b = 0.0562$$

# Vectors and Matrices

- Think of vectors as lists
- Think of matrices as arrays (lists of lists)

```
V1=[0 1 2 3 4]
V2=0:4
M1=[1 0 1;0 1 0;0 0 1]
M2=ones(3)
```

# Built-in Matrices

- **zeros(m,n)** – filled with 0's
- **ones(m,n)** – filled with ones
- **eye(n)** – identity
- **rand(n,m)** – random numbers
- **randn(n,m)** – normally distributed

# Accessing elements

- You can pick out individual components of vectors and matrices

**V1(3) – third element**

**M1(2,3) – row 2, column 3**

**M1(:,2) – all rows, column 2**

**M1(1,:) – row 1, all columns**

# Practice

- Generate vector (x) of positive integers less than 50
- Plot exp(x)
- Plot exp($x^2$)
- Plot exp(1/x)

**Create x vector**
**Then:**

```
y=exp(x)
plot(x,y)
```

# The "." operators

- Using a dot before an operator will force element by element math, as opposed to vector math

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$A * A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$A * A = \begin{bmatrix} 1*1+3*2 & 1*3+3*4 \\ 2*1+4*2 & 2*3+4*4 \end{bmatrix}$$

$$A * A = \begin{bmatrix} 7 & 15 \\ 10 & 22 \end{bmatrix}$$

$$A.* A = \begin{bmatrix} 1*1 & 3*3 \\ 2*2 & 4*4 \end{bmatrix}$$

$$A.* A = \begin{bmatrix} 1 & 9 \\ 4 & 16 \end{bmatrix}$$

# Vector Math

- Try this:

```
v=0:5
z=v*v
```

- To square each element:

```
z=v.*v
```

- Also ./ and .^

# Practice

- Try again to plot $\exp(x^2)$ for $0<x<50$

# Practice

- Use data below to find average and maximum speeds for the 5 routes
- (Divide distance by time, term-by-term)
- Average of vector is **mean(v)**
- Maximum is **max(v)**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Distance (mi)** | **560** | **440** | **490** | **530** | **370** |
| **Time (hr)** | **10.3** | **8.2** | **9.1** | **10.1** | **7.5** |

# Vector and Matrix Functions

- **length(A)** – length of vector
- **size(A)** – size of matrix
- **diag(A)** – diagonal of matrix
- **inv(A)** – inverse of matrix

# Functions

**exp, log, log10, sqrt**

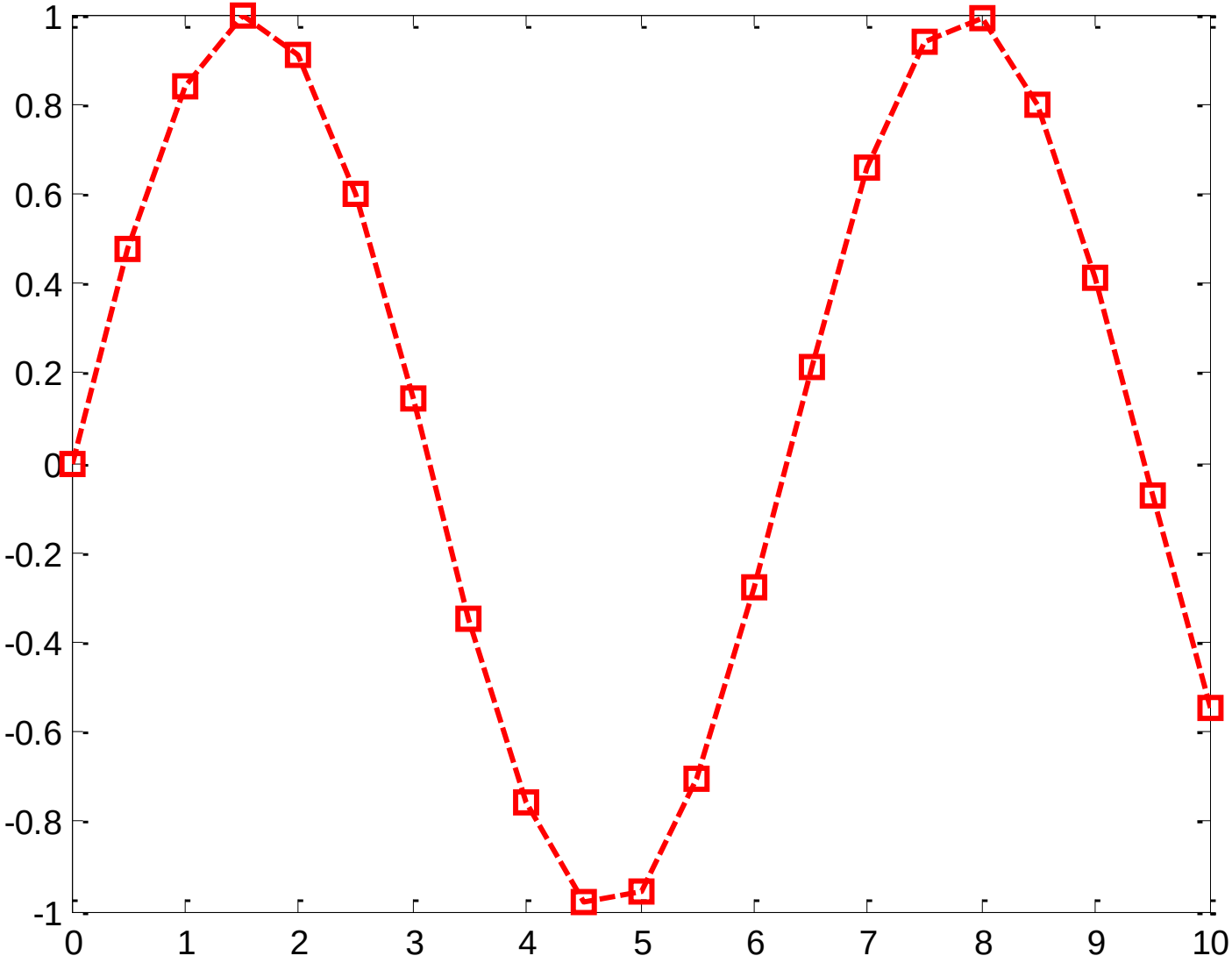
**sin, cos, tan, asin, acos, atan**

**max, min, mean, median, sum, prod, sort**

# Plotting – Again

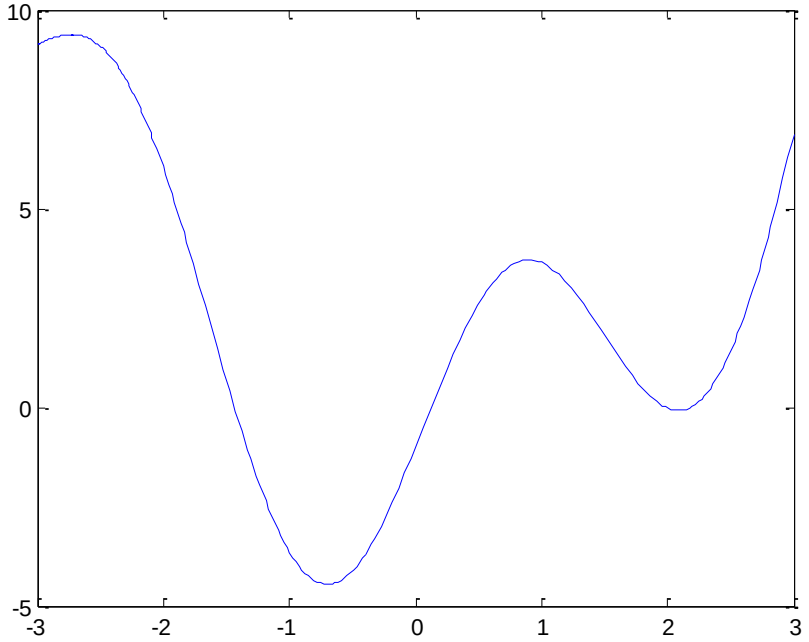
- Make vectors for x and y axis and then plot them

```
x=0:0.1:10
y=sin(x)
plot(x,y,'--rs','LineWidth',2)
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9
10

# More Graphics

**fplot(@(x) x^2+4*sin(2*x)-1,[-3,3])**



# More Graphics

- Adding axes, labels, and legends

```
xlabel('Time (seconds)')
legend('\alpha =1')
axis([0 3 -1  1])
```

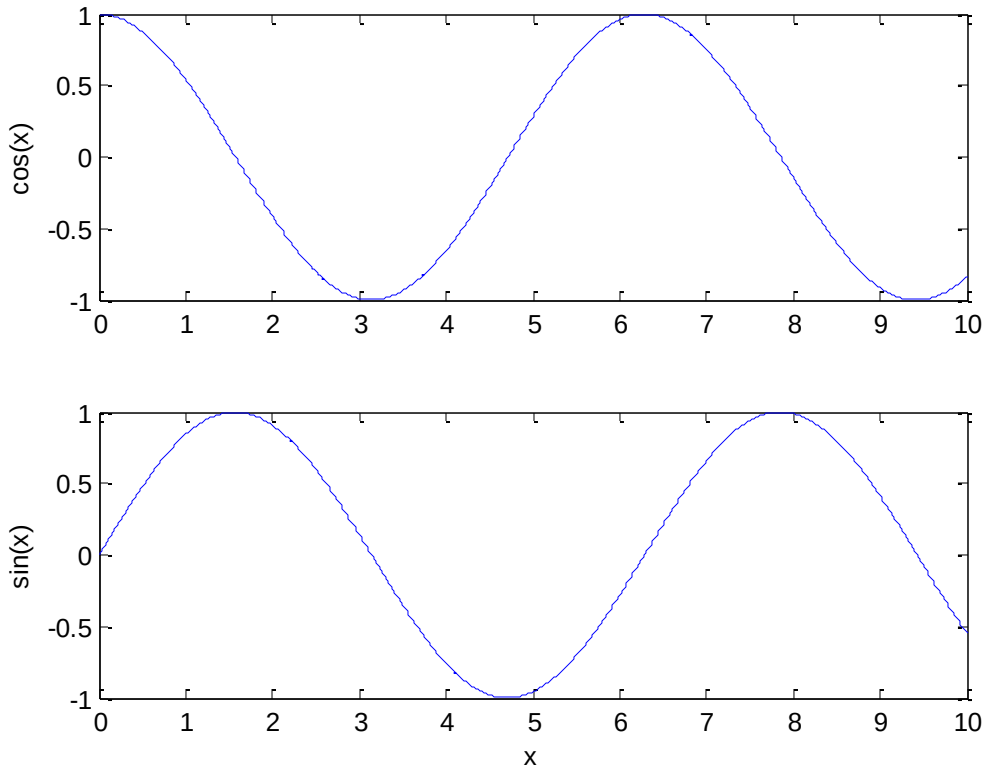
- Put this in after plot command
- Opening a new plot window

```
figure
```

# Practice

- Plot sin(1/x) from 0 to 0.2
- Put in labels, a title, and a legend

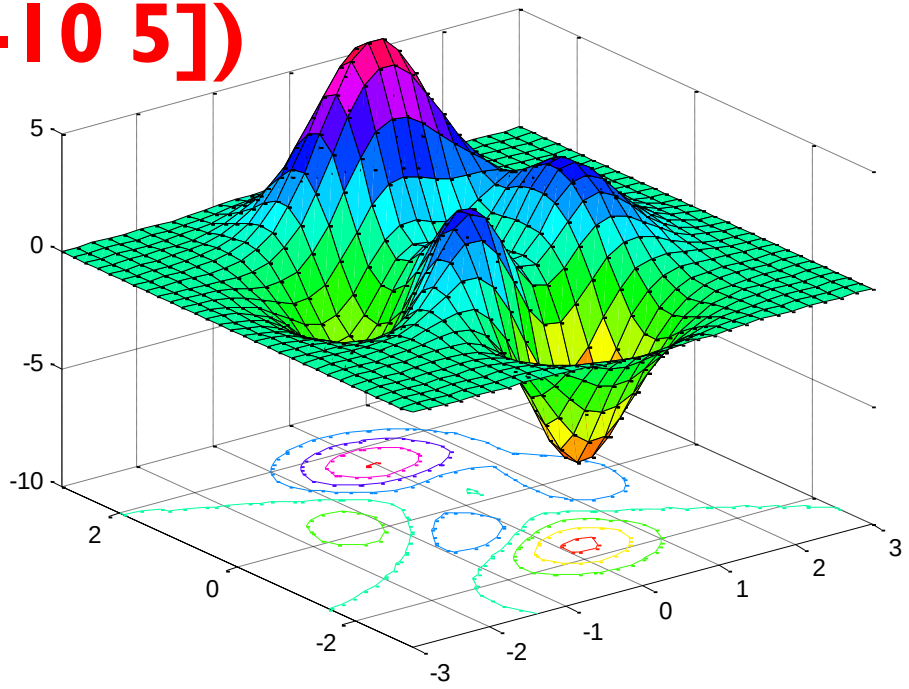
# Grouping Plots




```
x=0:0.01:10;
y=sin(x);
z=cos(x);
subplot(2,1,1)
plot(x,z)
ylabel('cos(x)');
subplot(2,1,2)
plot(x,y)
ylabel('sin(x)');
xlabel('x');
```

# Surface Plots

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z)
colormap hsv
axis([-3 3 -3 3 -10 5])
```



# Practice

- Plot $\cos(x^2+y^2)$ for $-3<x<3$ and $-3<y<3$
- Commands on next page will generate appropriate "z" matrix
- You just need to add in a **surfc(x,y,z)** command
- File ***ForSurfPlot.m*** will set up the matrices

# Generating Values for Surface Plot (we will learn this later)

```
N=100
lowX=-3
highX=3
for i=1:N
    for j=1:N
        x(i,j)=lowX+i*(highX-lowX)/N;
        y(i,j)=lowX+j*(highX-lowX)/N;
        z(i,j)=cos(x(i,j).^2+y(i,j).^2);
    end
end
```

# Other Plot Types

- Vertical bar
- Horizontal bar
- Stairs
- Stem
- Pie
- Histogram
- polar

# Adjusting Plots Interactively

- Demo

# Animation

- To animate a plot, simply generate a series of snapshots and then use "move" to show them
- Example, animate sin(x)*sin(2*pi*t/20)
- Get file ***anim.m***

# Animation Example

```
x=0:pi/100:2*pi;
y=sin(x);
plot(x,y)
axis tight
set(gca,'nextplot','replacechildren');
% Record the movie
for j = 1:20
    plot(x,sin(2*pi*j/20)*y)
    F(j) = getframe;
end
% Play the movie two times
movie(F,2)
```

# Questions?