Introduction to Matlab

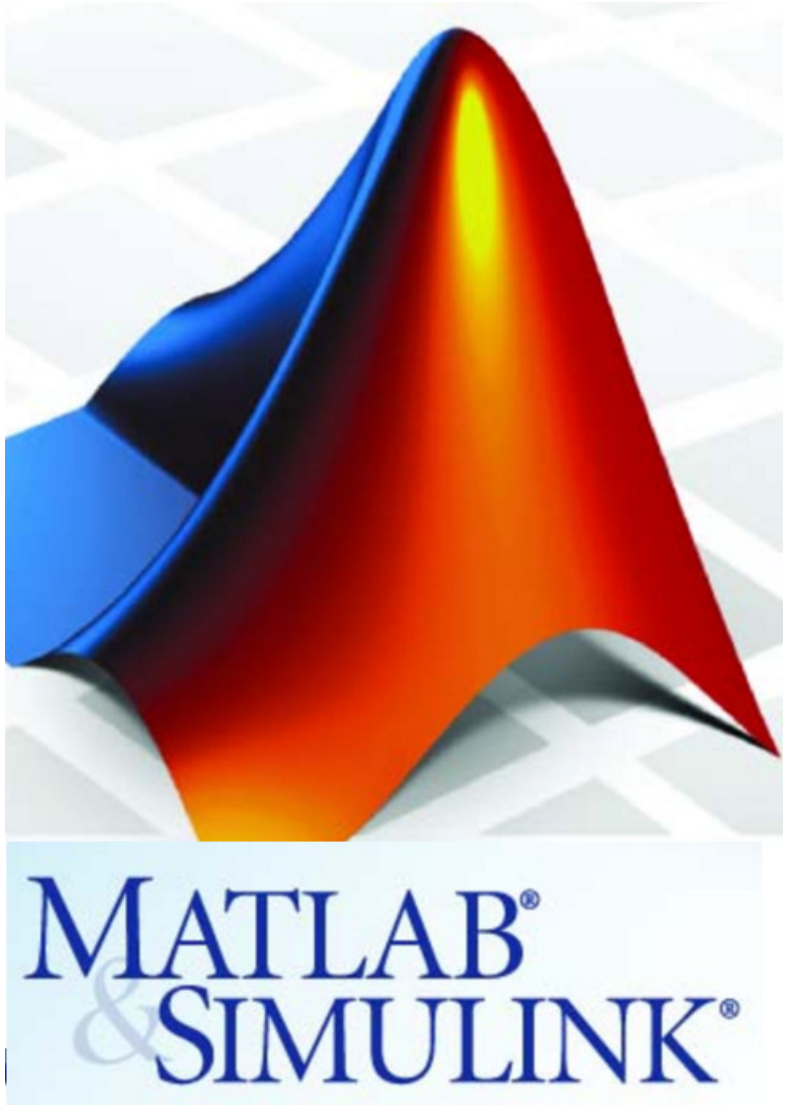
ECE 111 Introduction to ECE

Jake Glower - Week #1

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Becoming familiar with MATLAB

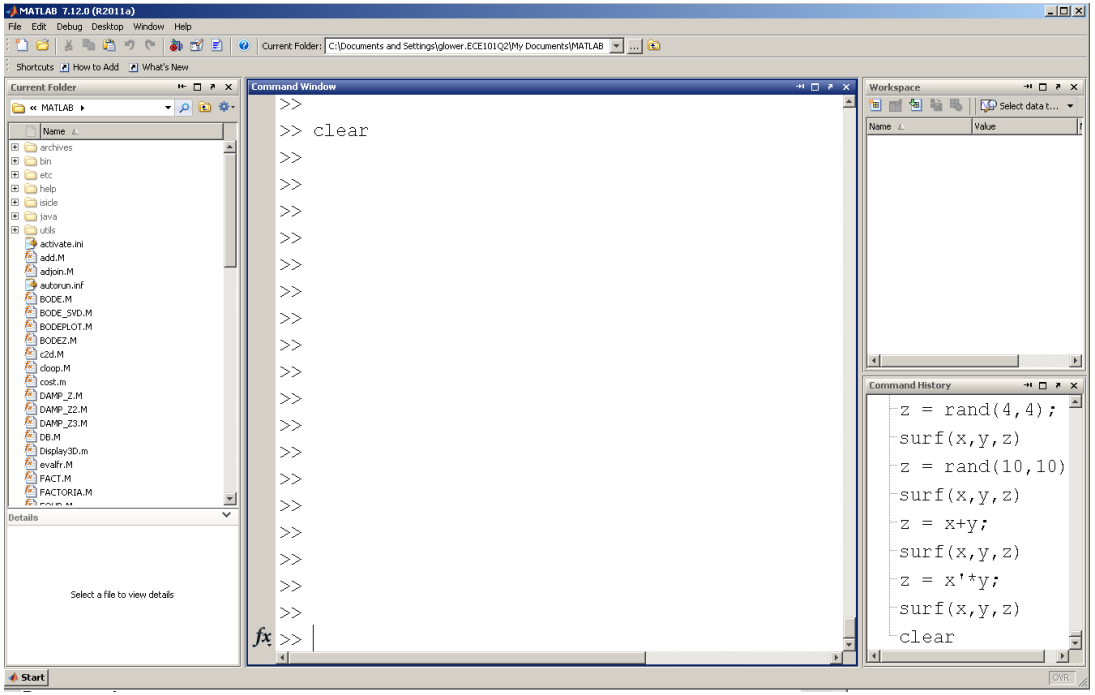
- The console
- The editor
- The graphics windows
- The help menu
- Random numbers in Matlab
- If-Statements
- For-Loops
- While-Loops
- Monte-Carlo Simulations



General environment and the console

Startup Screen:

- Looks like this
- I usually close everything down except the command window



Command Window

Matlab works like a calculator

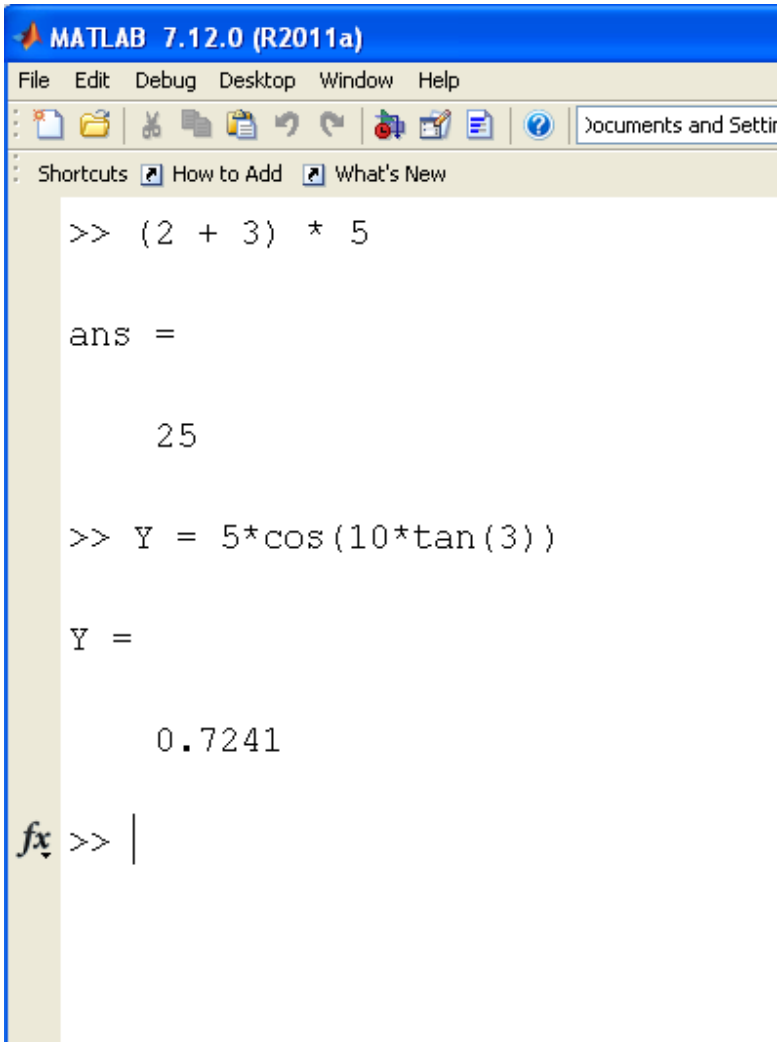
- Solve

```
( 2 + 3 ) * 5
```

- Solve

```
Y = 5*cos(10*tan(3))
```

You type it the way it looks:



Matrices in Matlab

Matlab is a matrix language

- An nxm matrix has n rows, m columns

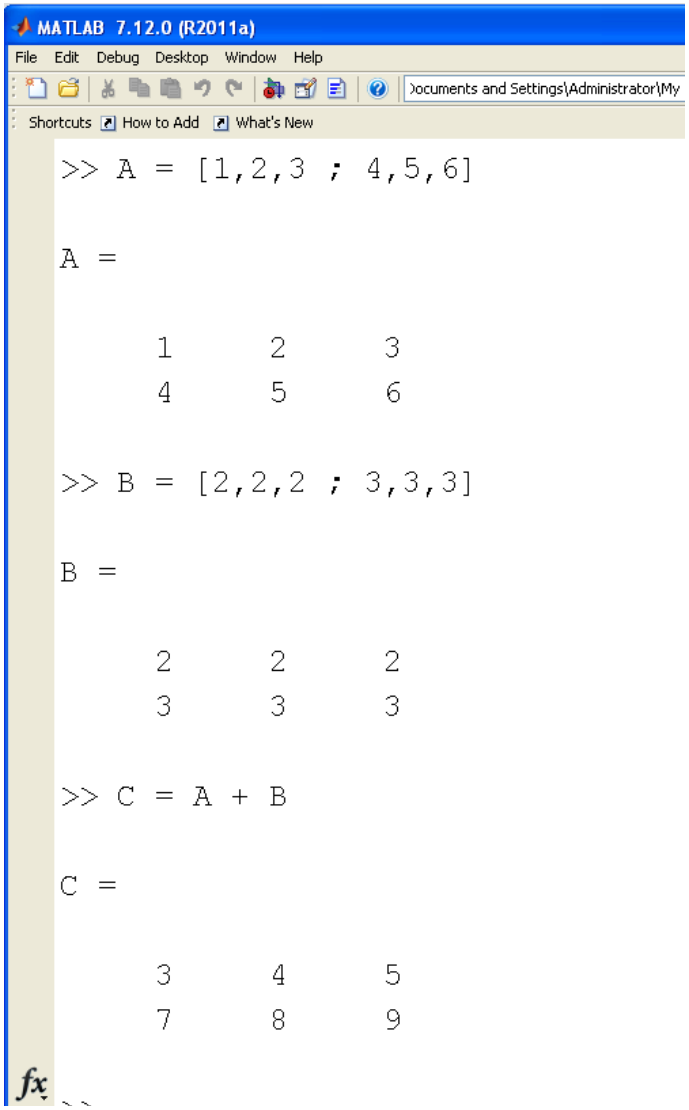
$$A_{2x3} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

The syntax to input a matrix are:

```
[       start of a matrix
,       next column (a space also works)
;       next row
]       end of matrix.
```

Addition: The dimensions must match

- 2x3 + 2x3 = 2x3



Multiplication:

The inner dimension must match

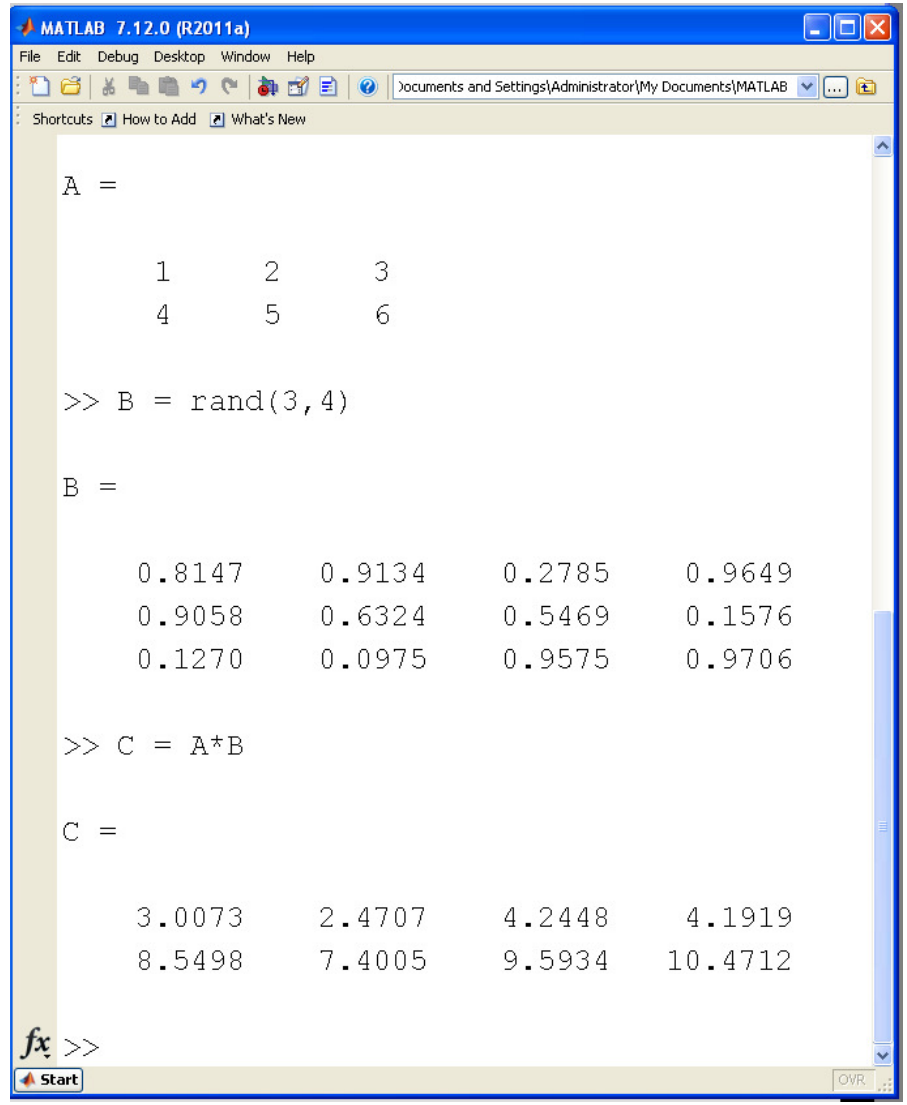
- 2x3 * 3x4 = 2x4

$$C_{mn} = A_{mx} B_{xn}$$

Element (i,k) of matrix C is computed as:

$$c_{ik} = \Sigma a_{ik} b_{kj}$$

Note that matrix multiplication is *not* commutative

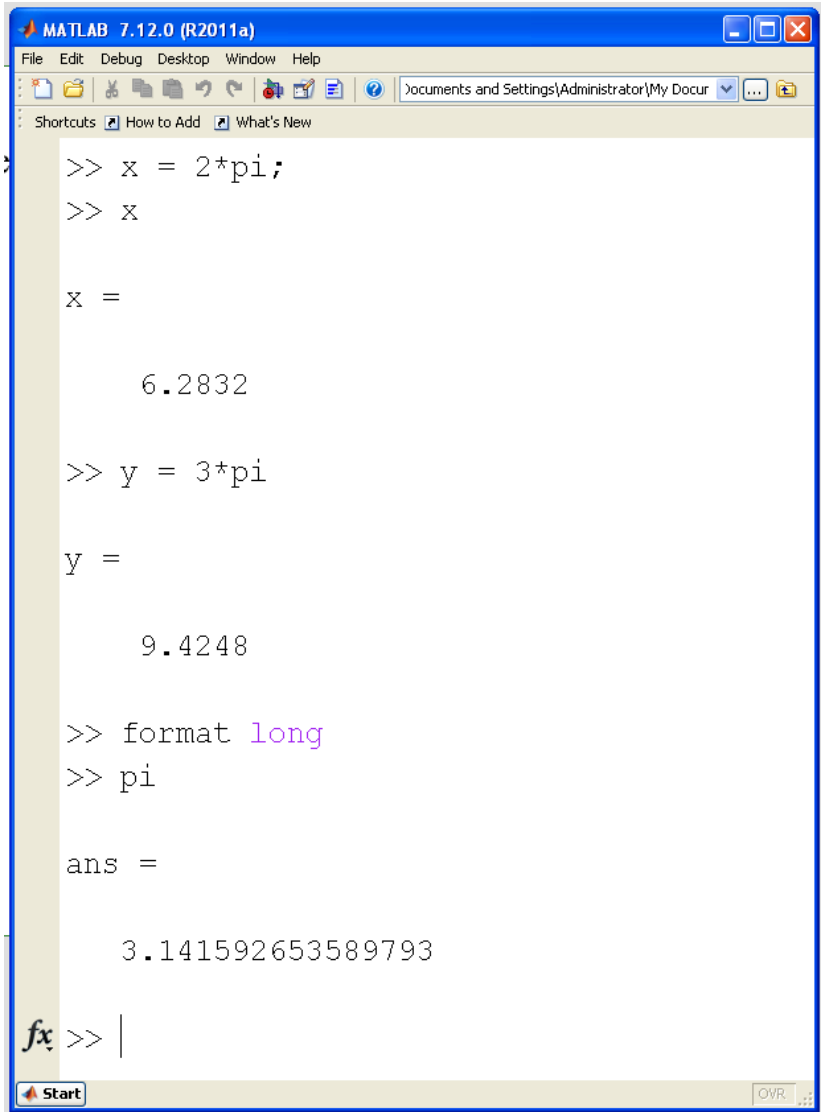


Formatting Output

Terminate a line of you don't want the result displayed

- It's still computed

```
format short
pi

   3.1416

format long
pi

  3.141592653589793


format shorteng
pi^30

  821.2893e+012
```

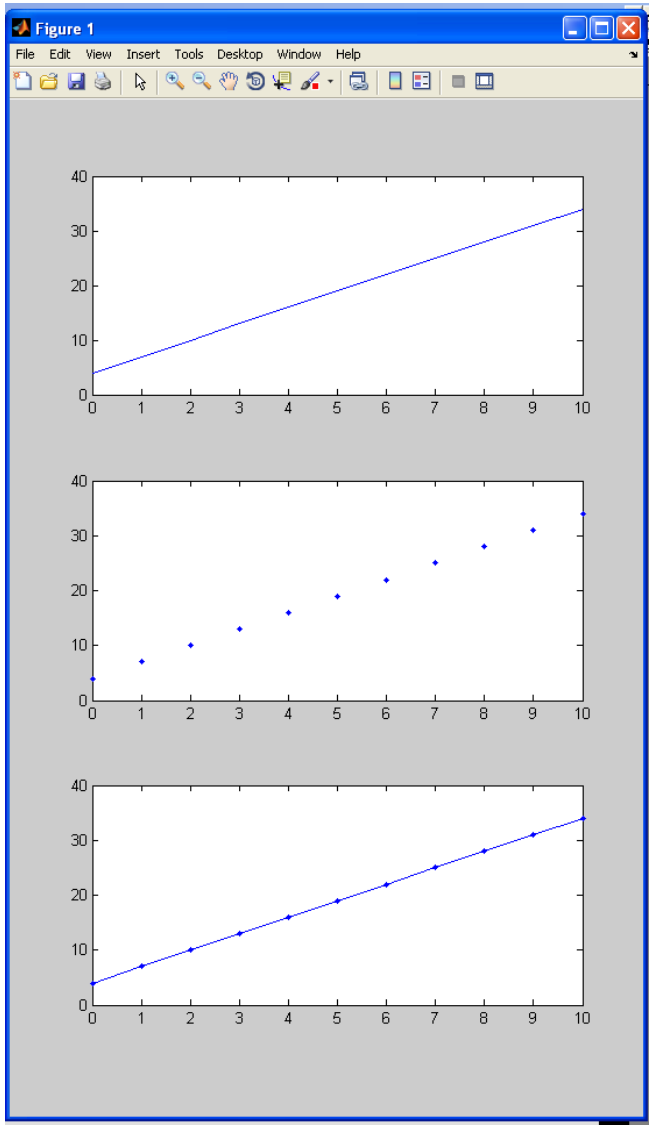


Plotting Functions in Matlab:

Matlab has some pretty good graphics capabilities.

Matlab Plot Command	x axis	y axis	type of function
plot(x,y)	linear	linear	$y = ax + b$
semilogx(x,y)	log()	linear	$y = a\log(bx)$
semilogy(x,y)	linear	log()	$y = a\,e^{bx}$
loglog(x,y)	log()	log()	$y = a \cdot b^{x}$
subplot(abc)	Create 'a' rows, 'b' columns of graphs. Starting at #c		

```
x = [0:1:10]';
y = 3*x + 4;

subplot(311)
plot(x,y);
subplot(312)
plot(x,y,'.');
subplot(313);
plot(x,y,'.-');
```

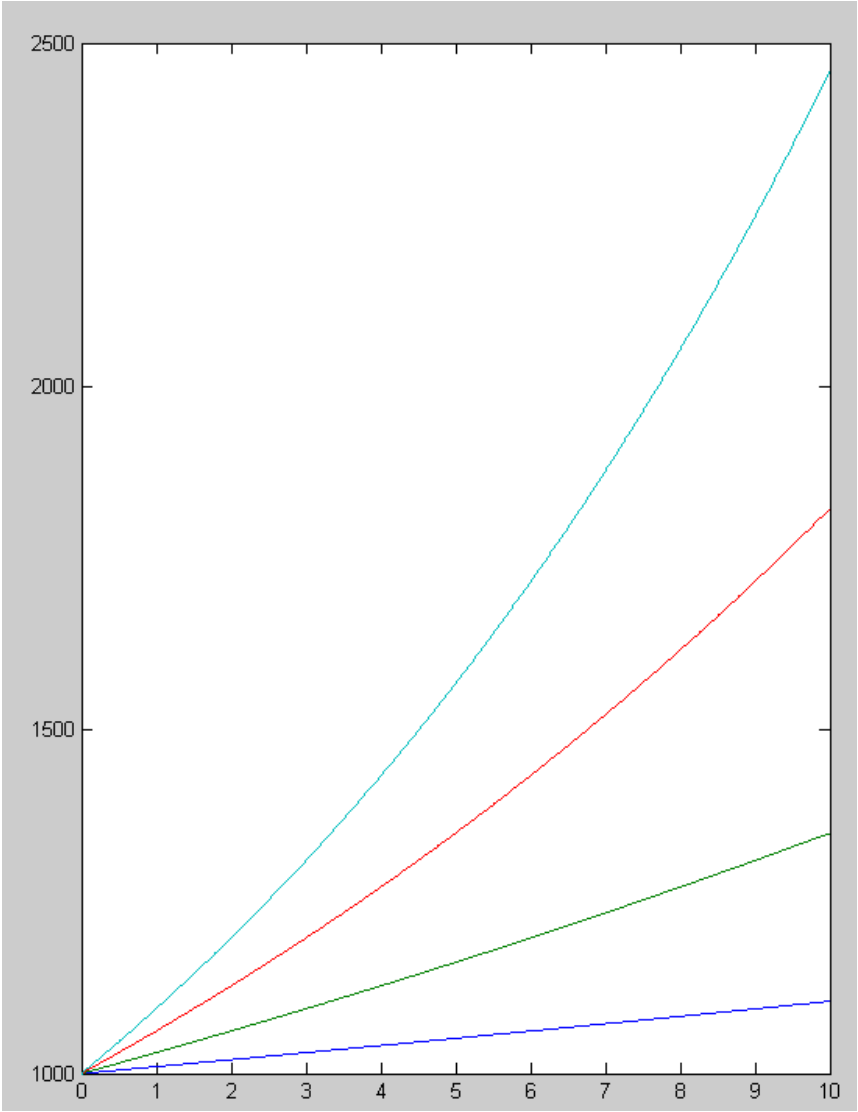


Multiple Plots on the same graph:

Invest $1000 for 10 years at...

- 1% interest
- 3% interest
- 6% interest
- 9% interest

```
t = [0:0.01:10]';
y1 = 1000 * exp(0.01*t);
y3 = 1000 * exp(0.03*t);
y6 = 1000 * exp(0.06*t);
y9 = 1000 * exp(0.09*t);

% Method #1
plot(t,y1,t,y3,t,y6,t,y9)

% Method #2
plot(t,[y1,y3,y6,y9])
```



Polynomials

poly([a,b,c])

- Give a polynomial with roots at (a, b, c)

roots([a,b,c,d])

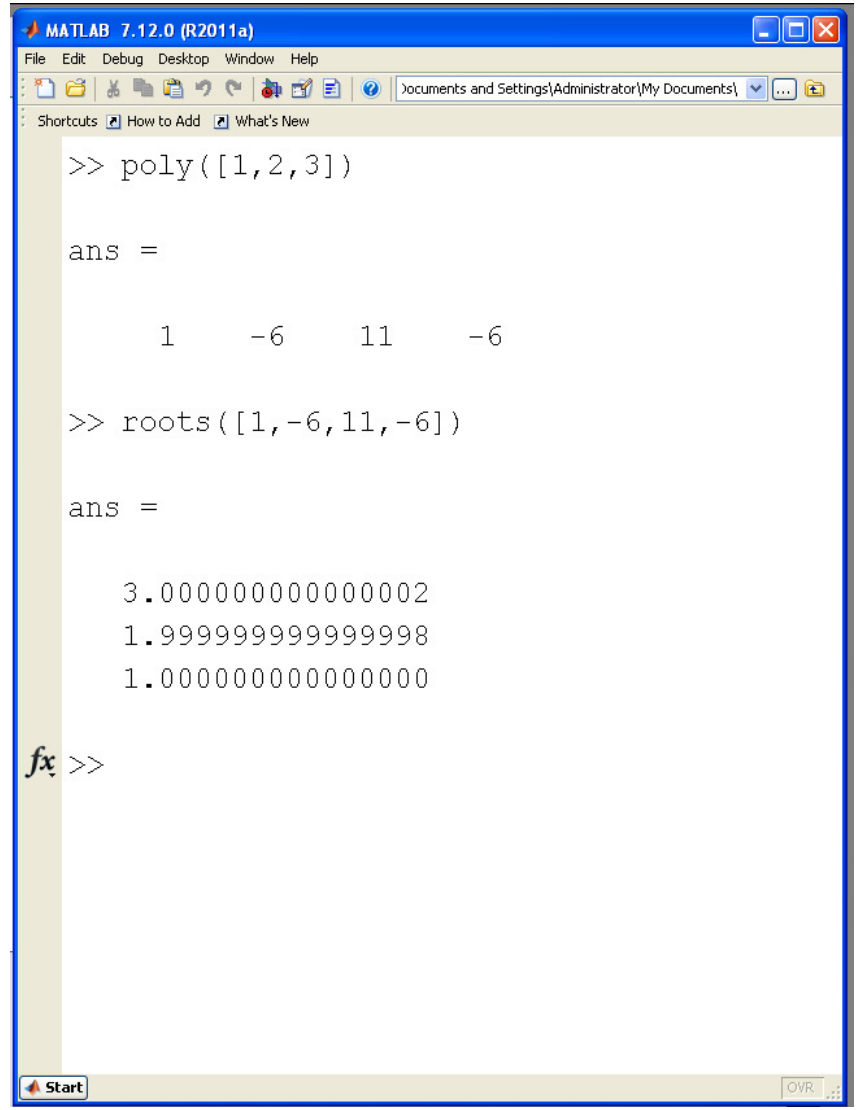
- Find the roots of the polynomial

$$ax^3 + bx^2 + cx + d = 0$$

```
poly([1,2,3])

     1    -6    11    -6
```

$$y = x^3 - 6x^2 + 11x - 6$$
$$= (x-1)(x-2)(x-3)$$

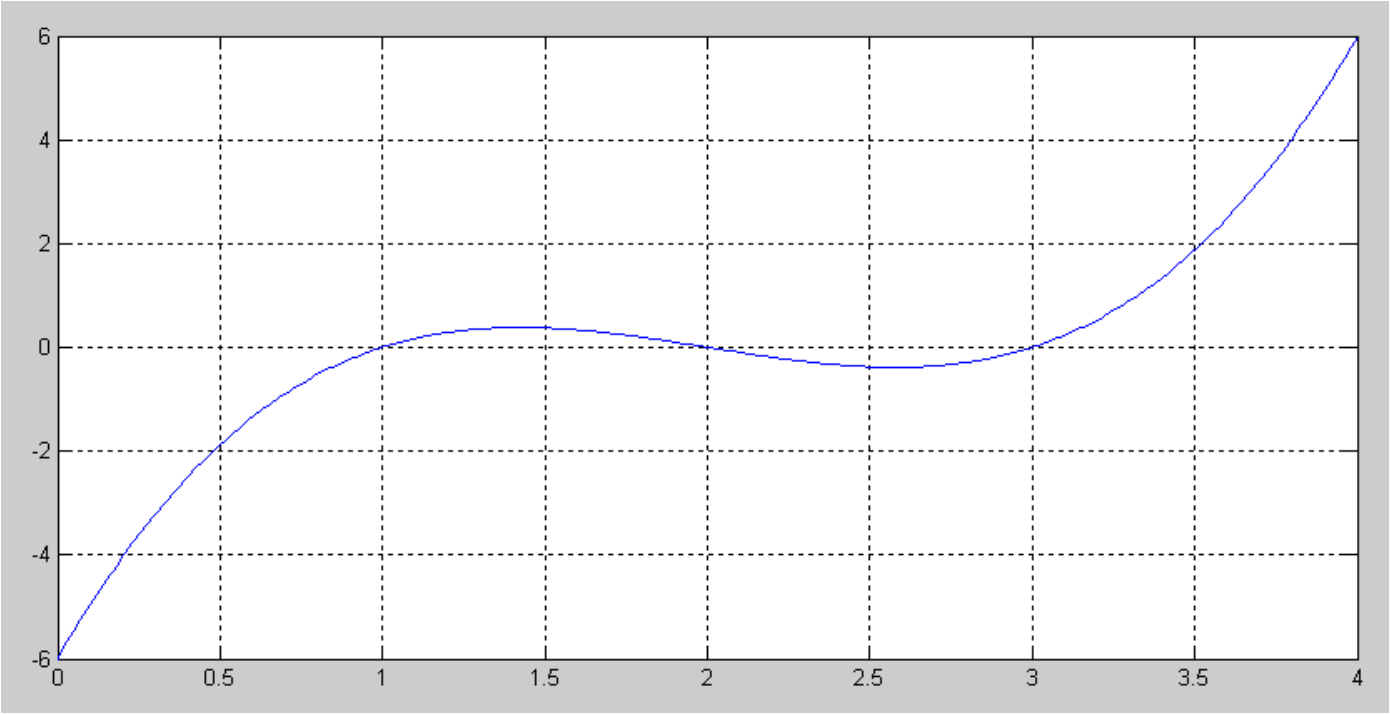
```
roots([1,-6,11,-6])

   3.0000
   2.0000
   1.0000
```



Note: The roots are the zero crossings

- Roots = { 1, 2, 3 }

```
x = [0:0.01:4]';
y = x.^3 - 6*(x.^2) + 11*x - 6;
plot(x,y);
grid on
```



3-D Plots

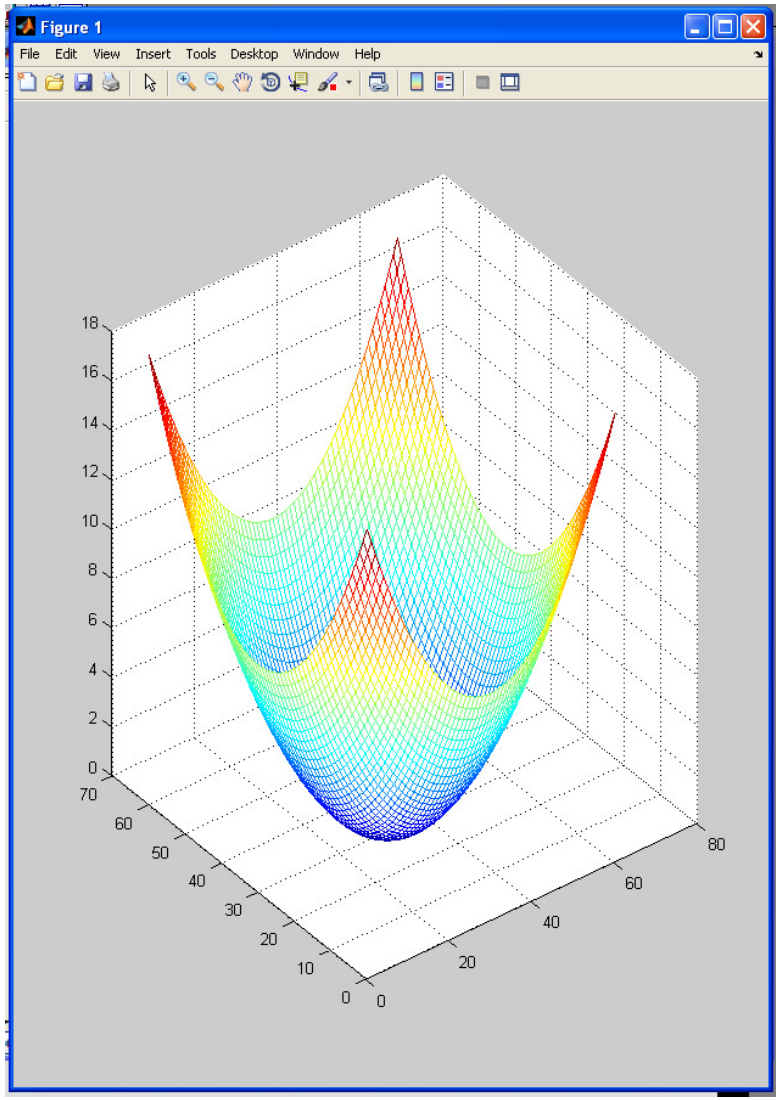
```
mesh(z)
```

Draw a 3-d mesh for the array 'z' with the height being the value in the array.

Example

$$z = x^2 + y^2$$

```
x = [-3:0.1:3]';
y = [-3:0.1:3]';
z = zeros(61,61);
for i=1:61
  for j=1:61
     z(i,j) = x(i)^2 + y(j)^2;
     end
  end

mesh(z)
```

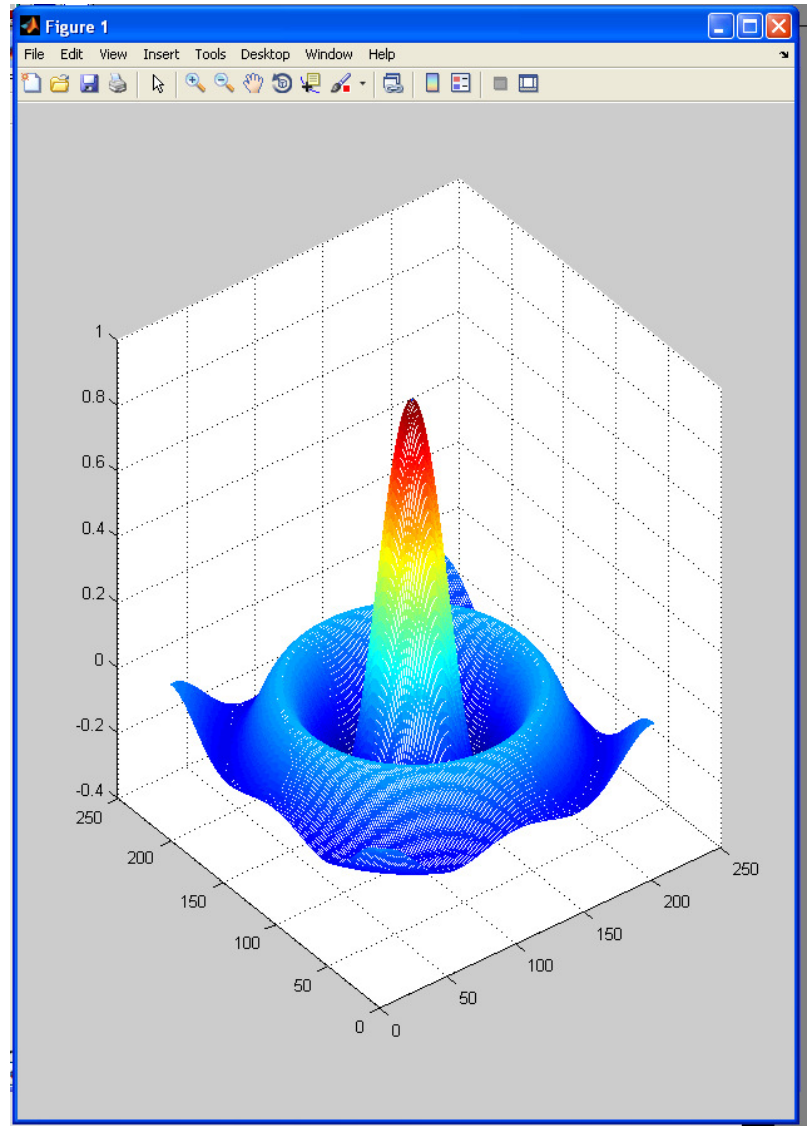


Another pretty plot:

$$r = \sqrt{x^2 + y^2}$$

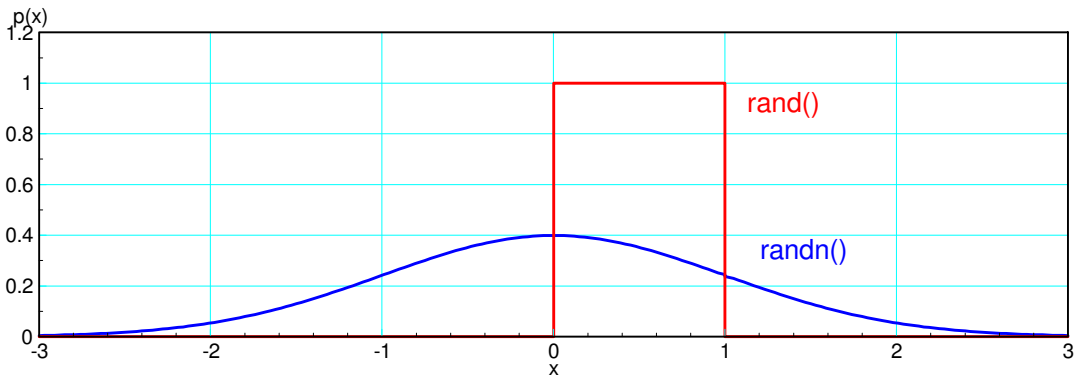
$$z(x, y) = \left(\frac{\sin(r)}{r} \right)$$

```
x = [-10:0.1:10]';
y = [-10:0.1:10]';
z = zeros(201,201);
for i=1:201
  for j=1:201
     r = sqrt(x(i)^2 + y(j)^2);
     z(i,j) = sin(r) / (r + 0.000001);
     end
  end
mesh(z)
```

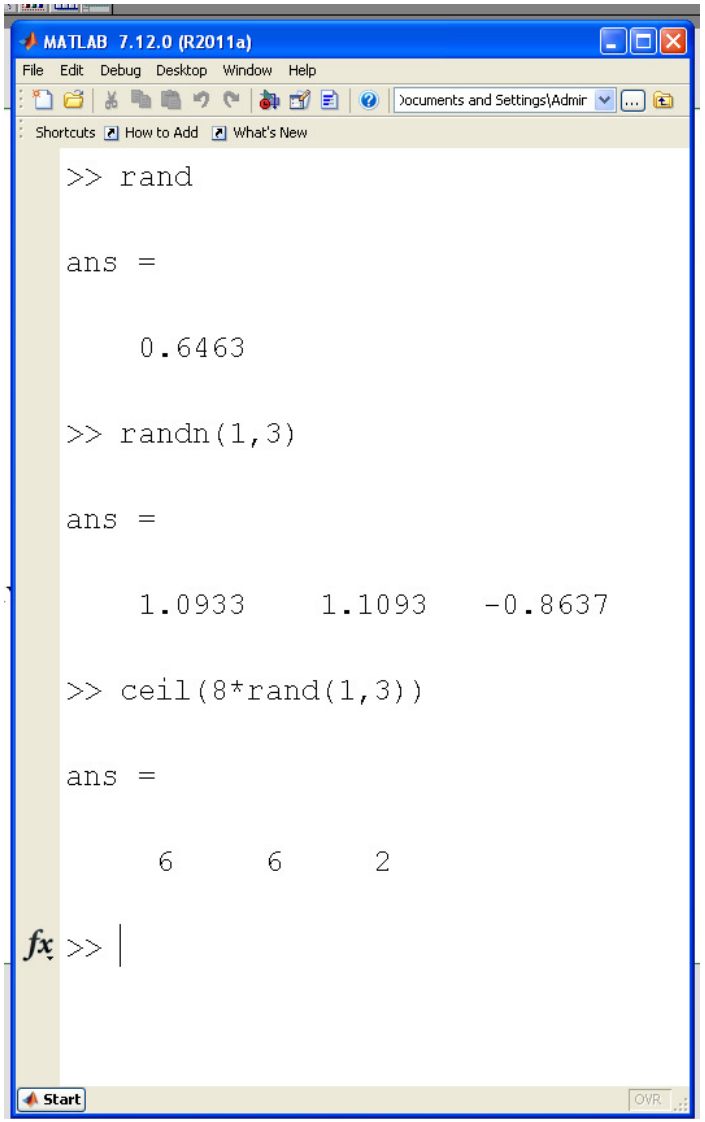


Random Numbers: Rolling Dice

```
rand          random number: (0,1)
randn         standard normal random #
```

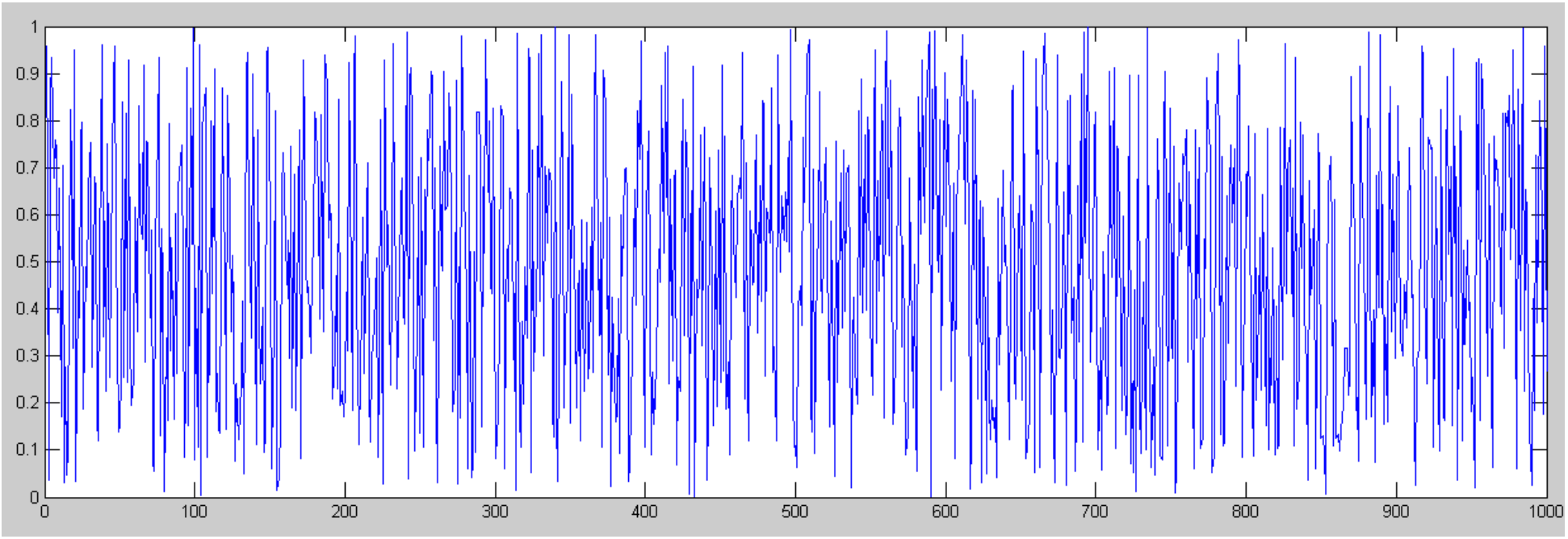



```
rand(1,5)                1x5 matrix of random #

ceil( 6*rand )           6-sided die

ceil( 8*rand(1,3) )      3d8

sum( 6*rand(5,1) )       sum of 5d6
                         (level 5 fireball)
```



Generate 1000 random numbers and plot them

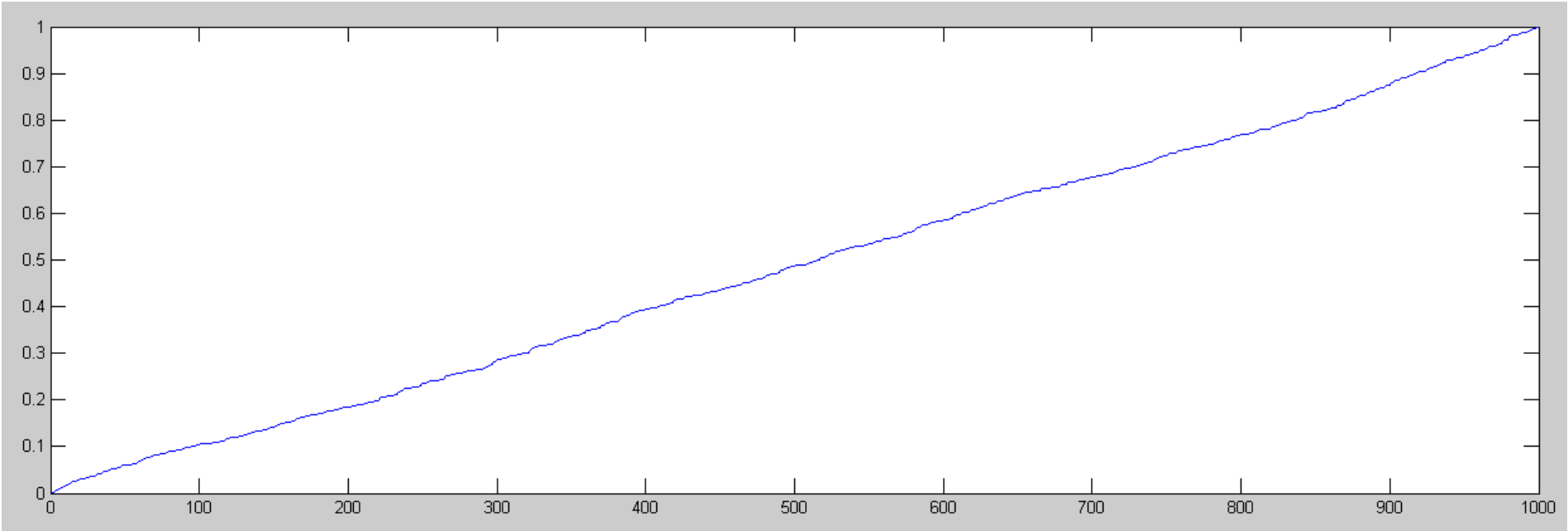
```
x = rand(1,1000);
plot(x)
```



If you sort the numbers, you can see the cumulative distribution function

- approximately
- (rand < 0.6) approximately 60% of the time

```
y = sort(x);
plot(y)
```



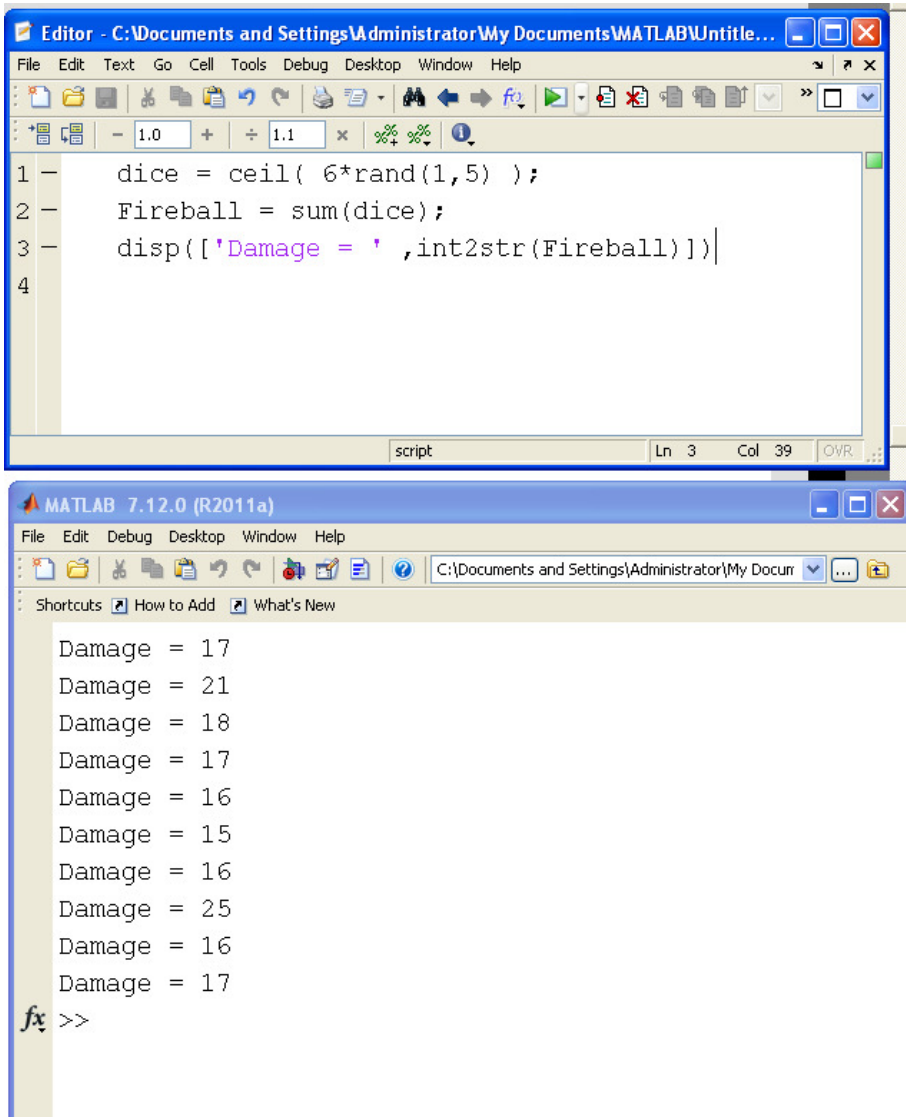
Matlab Scripts

Instead of typing in the command window, you can use a Matlab Script

When executed, it acts like you input those instructions in the command window.

This is useful

- It lets you build up your code
- It lets you modify existing code
- You can rerun the code over and over



For-Loops

```
for i=1:100
    statements
    end
```

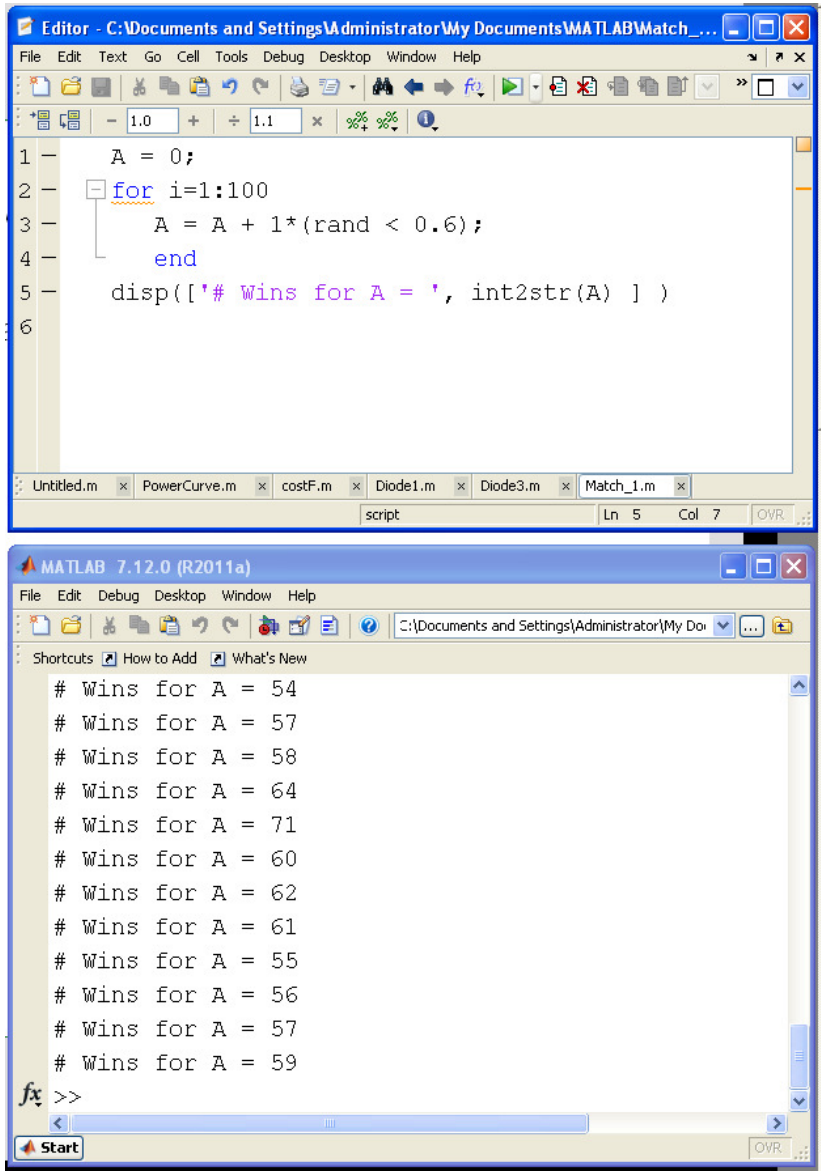
For-Loops repeat a set of instructions a fixed number of times

Example: A and B are playing a game.

- A has a 60% chance of winning any given game.
- How many games will A win in a 100 game match?

Note:

- 0 is false, 1 is true
- On average, A wins 60 games out of 100
- In any given match, A could win 0..100 games



For-Loops (cont'd)

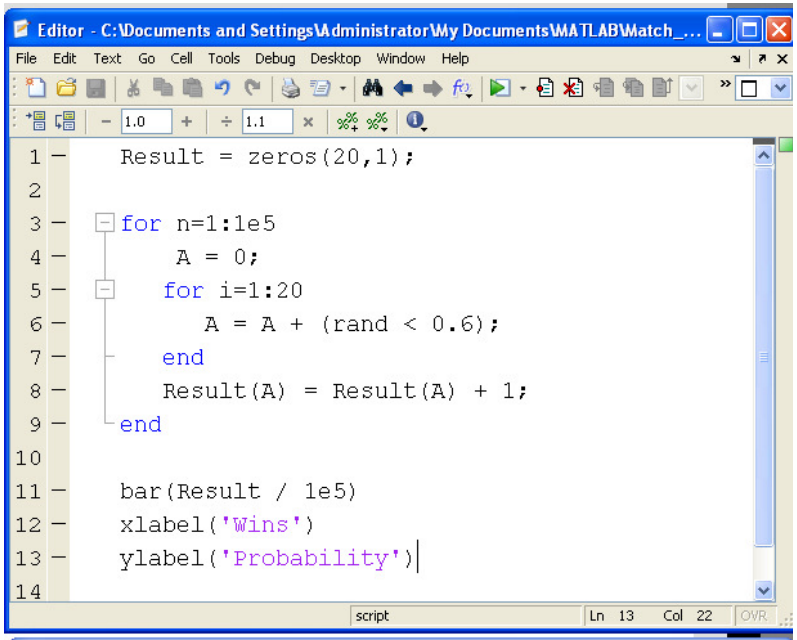
What is the probability of A winning X games in a 20 game match?

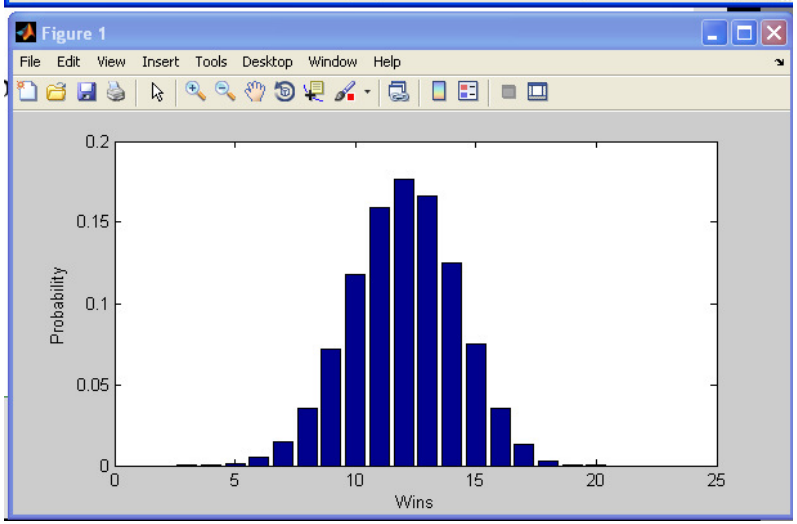
Solution:

- Play 100,000 matches.
- Count how many times A wins X games

Note

- The result is bell-shaped curve
- This is called a *Normal Distribution*
- You'll see this shape over and over again





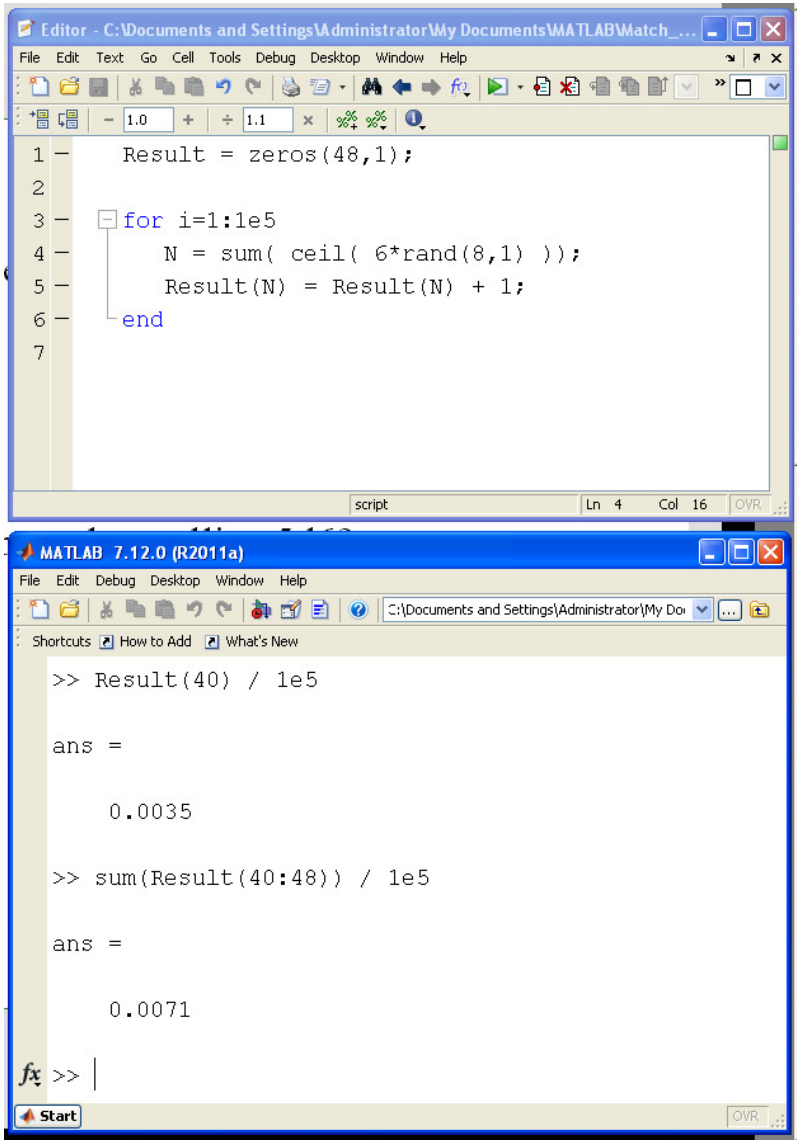
For-Loops (cont'd)

What is the probability of doing 40 damage with a level-8 fireball?

What is the probability of doing 40+ damage with a level-8 fireball?

Solution:

- Cast a level-8 fireball
- Repeat 100,000 times
- Record how many times the result was N



If-Statements

```
if(boolean statement)
   statements to execute
   end
```

If-statements allow you to execute a set of instructions based upon a condition

Valid boolean statements:

`(N == 3)`	$N = 3$
`(N > 3)`	$N > 3$
`(N >= 3)`	$N \geq 3$
`(N != 3)`	$N \neq 3$
`(N >= 3)*(N <= 7)`	$3 \leq N \leq 7$
`((N < 3) | (N > 7))`	$(N < 3)\ or\ (N > 7)$

If-Statements (cont'd)

A and B are playing a match

- Each match consists of 5 games
- A has a 60% chance of winning any given game
- Whoever wins 3+ games wins the match

What is the chance that A wins the match?

Solution:

- Monte-Carlo simulation of 100,000 matches
- A wins about 68% of the time

Editor - C:\Documents and Settings\Administrator\My Documents\MATLAB\Match_...

```
Wins = 0;
for n=1:1e5
    A = 0;
    for i=1:5
        if(rand < 0.6)
            A = A + 1;
        end
    end
    if(A >= 3)
        Wins = Wins + 1;
    end
end
disp(['# of Wins for A = ', int2str(Wins)]);
```

MATLAB 7.12.0 (R2011a)

```
# of Wins for A = 68242
# of Wins for A = 68149
# of Wins for A = 68087
# of Wins for A = 68271
# of Wins for A = 68173
# of Wins for A = 68497
# of Wins for A = 68090
# of Wins for A = 68059
# of Wins for A = 68344
>>
```

If-Statements (cont'd)

Player A casts a level-8 fireball (8d6)

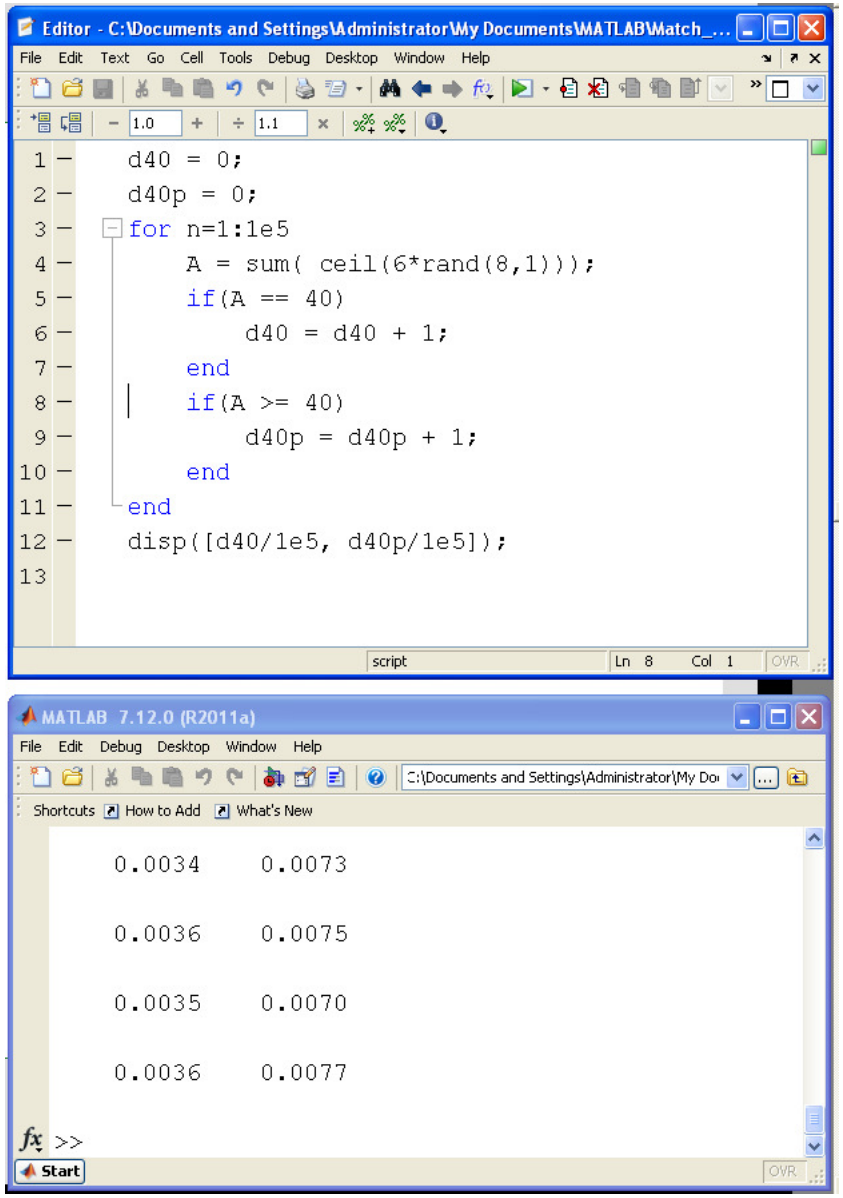
- What is the chance of doing 40 damage?
- What is the chance of doing 40+ damage?

Solution:

- Monte-Carlo Simulation
- Cast 100,000 fireballs
- Count how many times you do 40 damage
- Count how many times you do 40+ damage

Note:

- There are multiple ways to solve any given problem



If-Elseif-Else Statements

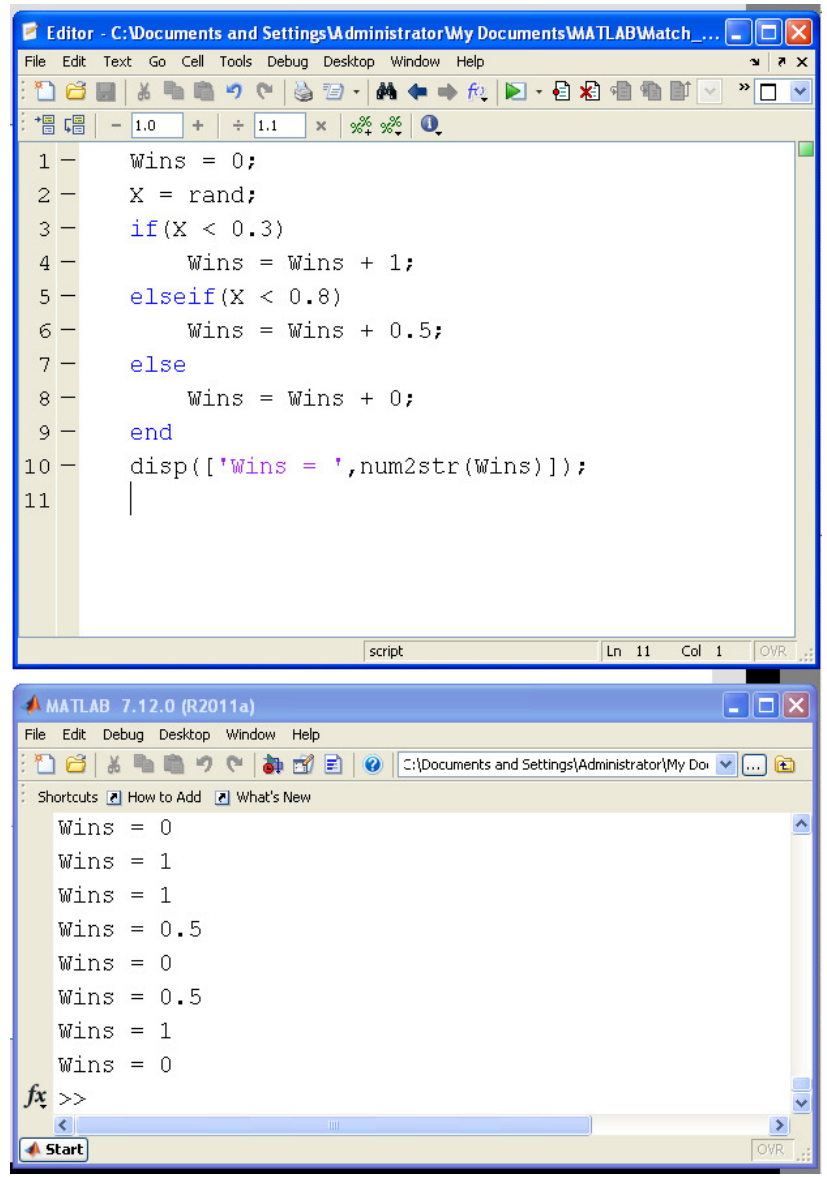
```
if(boolean statement)
   instrucitons #1
elseif(boolean statement)
   instructions #2
else
   instuctions #3
end
```

Execute a set of statements

- If the first condition is true, execute instructions #1 and skip the rest
- If the first condition is false , then check the second boolean statement
- If that is also false, excute instructions #3

Example: A and B play a game

- A has a 30% chance of winning (+1 point)
- A has a 50% chance of a tie (+1/2 point)
- A has a 20% chanec of losing (0 points)



If-Else Statements (cont'd)

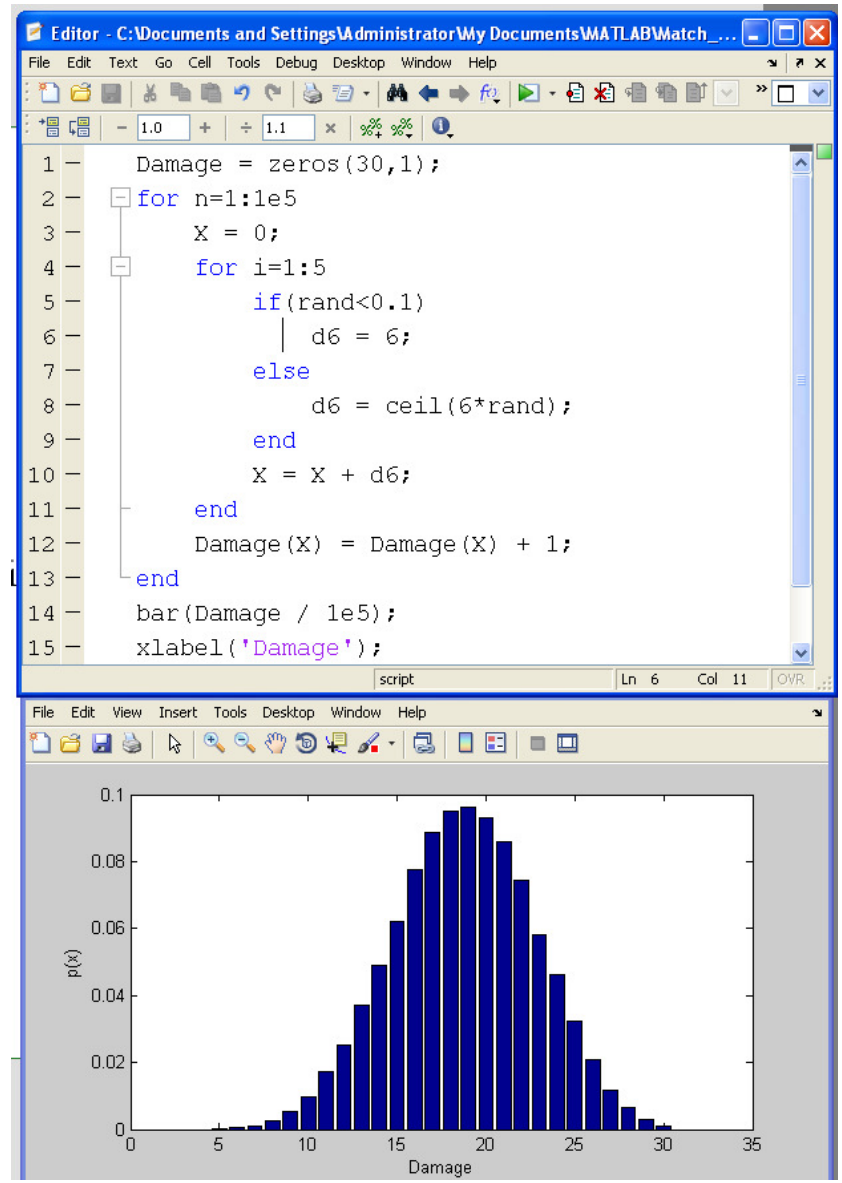
A die is loaded:

- 10% of the time, it always rolls a 6
- The rest of the time it's a fair die

What is the chance of doing X damage with a level-5 fireball?

Note:

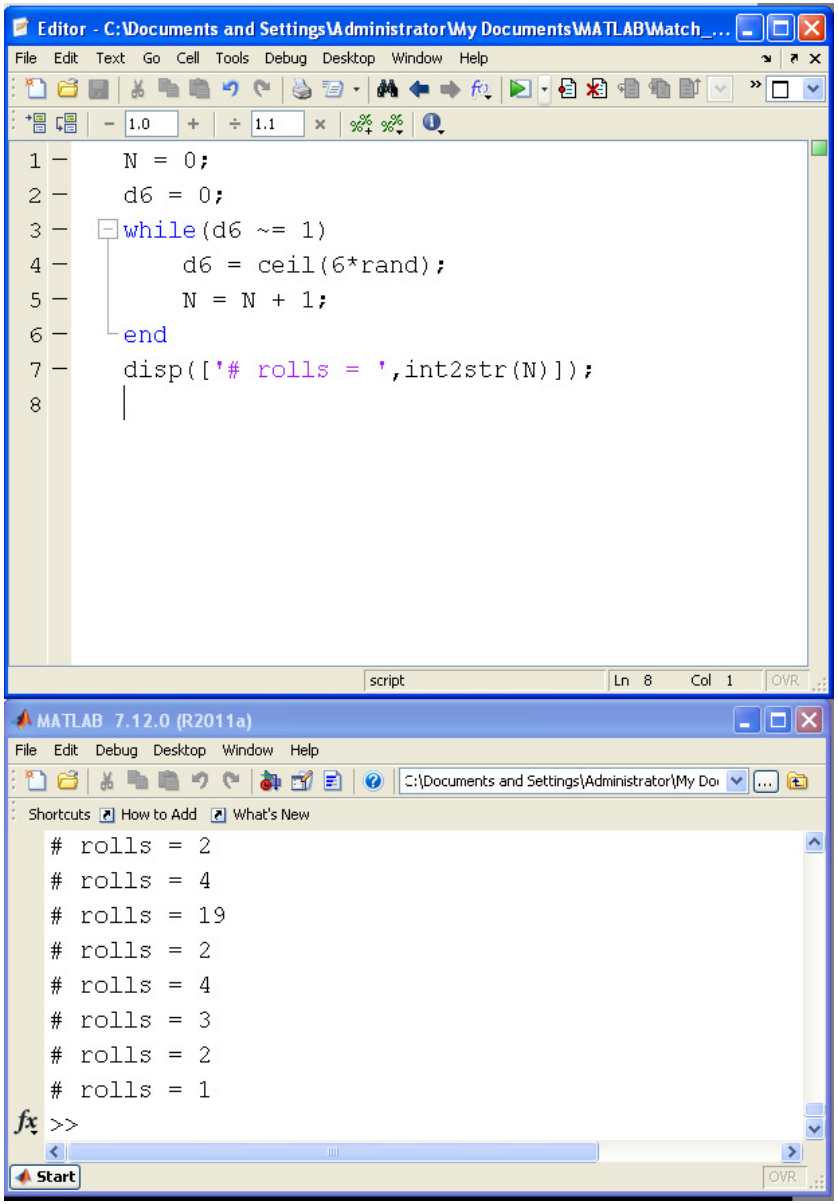
- Once again, the result is a bell-shaped curve
- This is *The Central Limit Theorem*
- Almost all distributions converge to this shape



While-Loop

```
while(statement is true)
  do the following
  end
```

Example: Count how many times you roll a die until you get a 1



While-Loop (cont'd)

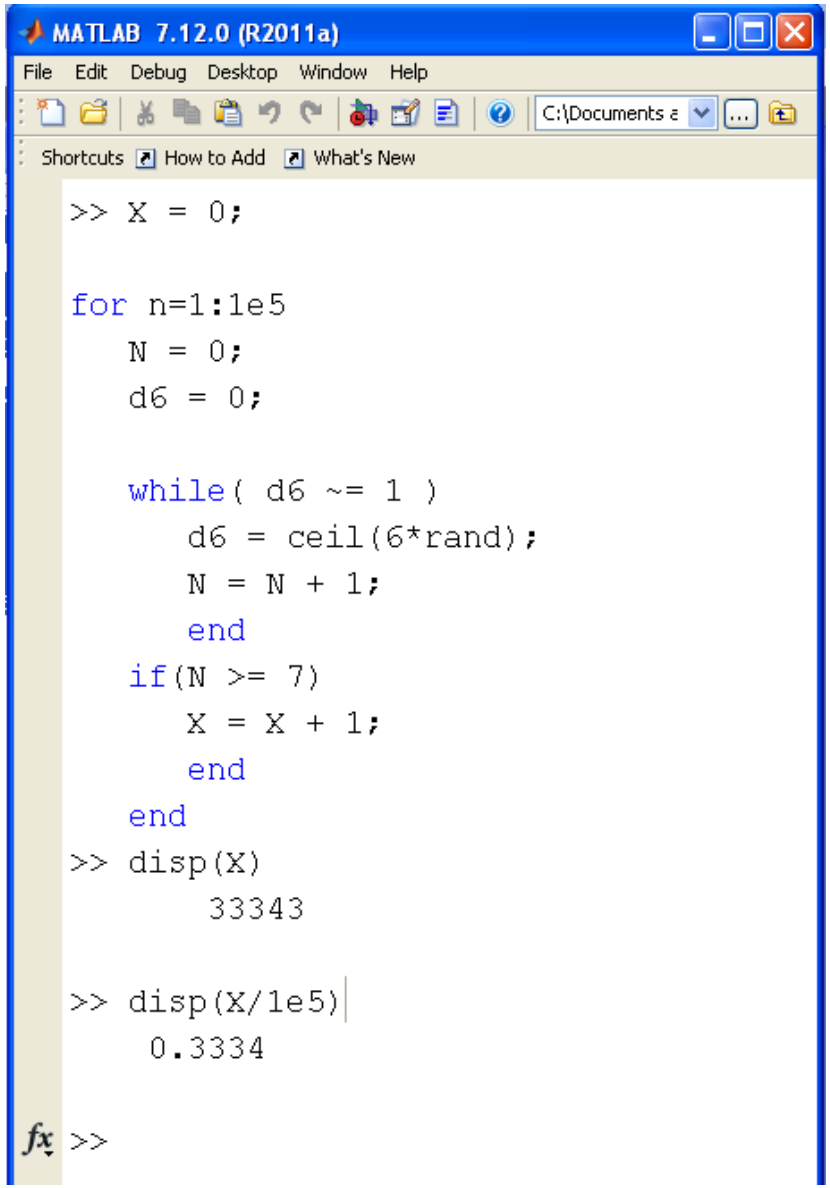
What is the chance that it will take 7 or more rolls to get a 1?

Repeat 100,000 times

- Monte-Carlo Simulation

In 100,000 trials

- It took 7 or more rolls 33,343 times
- There is about a 33.34% chance it will take 7 or more rolls to get a 1



While-Loop (cont'd)

Player A and B are playing a match

- Player A has a 60% chance of winning any given game
- When a player is up 3 games, the match is over

What is the chance player A wins the match?

Solution: Play a single match

- If A wins, A gains 1 point.
- If A loses, A loses 1 point.
- Keep playing until A is up 3 or down 3

Editor - C:\Documents and Settings\Administrator\My Documents\MATLAB\Match_...

```
A = 0;
while(abs(A) < 3)
    if(rand < 0.6)
        A = A + 1;
    else
        A = A - 1;
    end
end
if(A == 3)
    Wins = 1;
else
    Wins = 0;
end
disp(['Wins = ',int2str(Wins)])
```

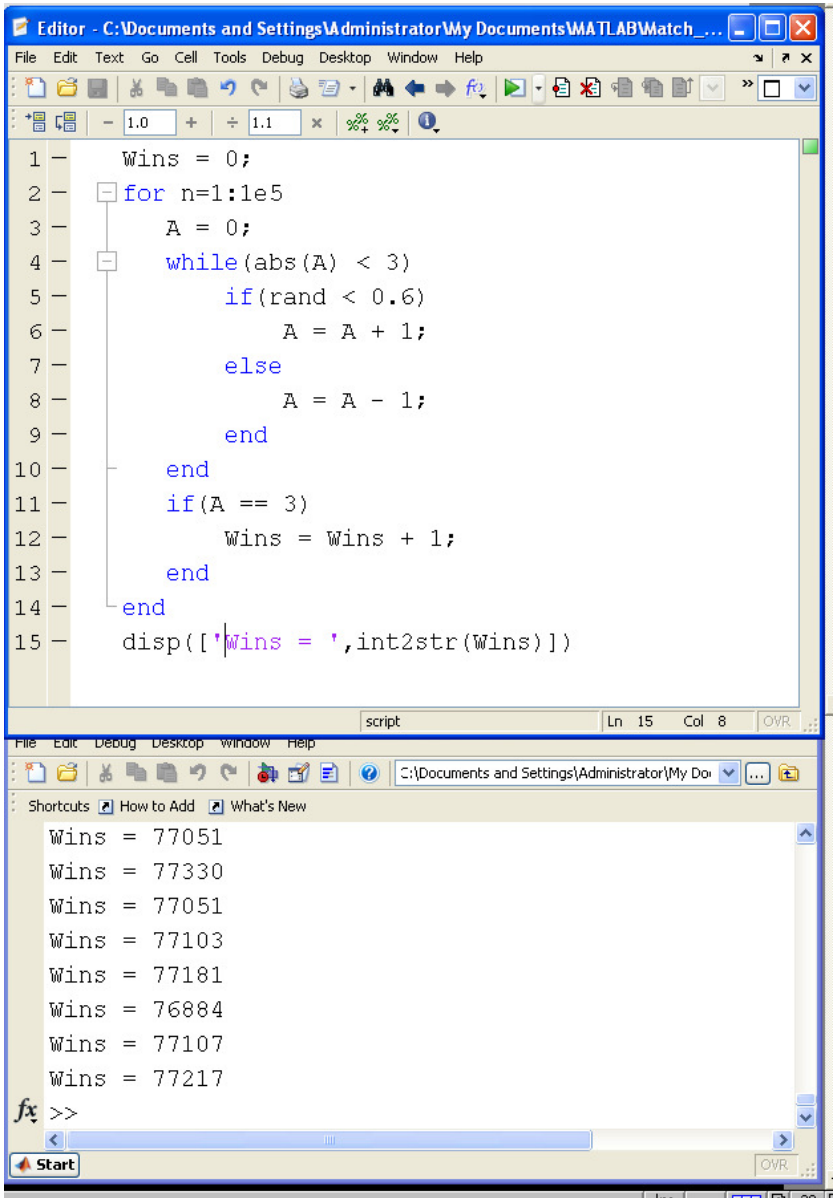
MATLAB 7.12.0 (R2011a)

```
Wins = 1
Wins = 0
Wins = 0
Wins = 1
Wins = 1
Wins = 1
Wins = 0
Wins = 0
>>
```

Win-by-3 (cont'd)

Now play 100,000 matches

- A wins about 77% of the time with this format
- TV hates this format since a match can take a very long time



Sort Command

Sort data in either

- Ascending order (default), or
- Descending order

a = sort(Deck);

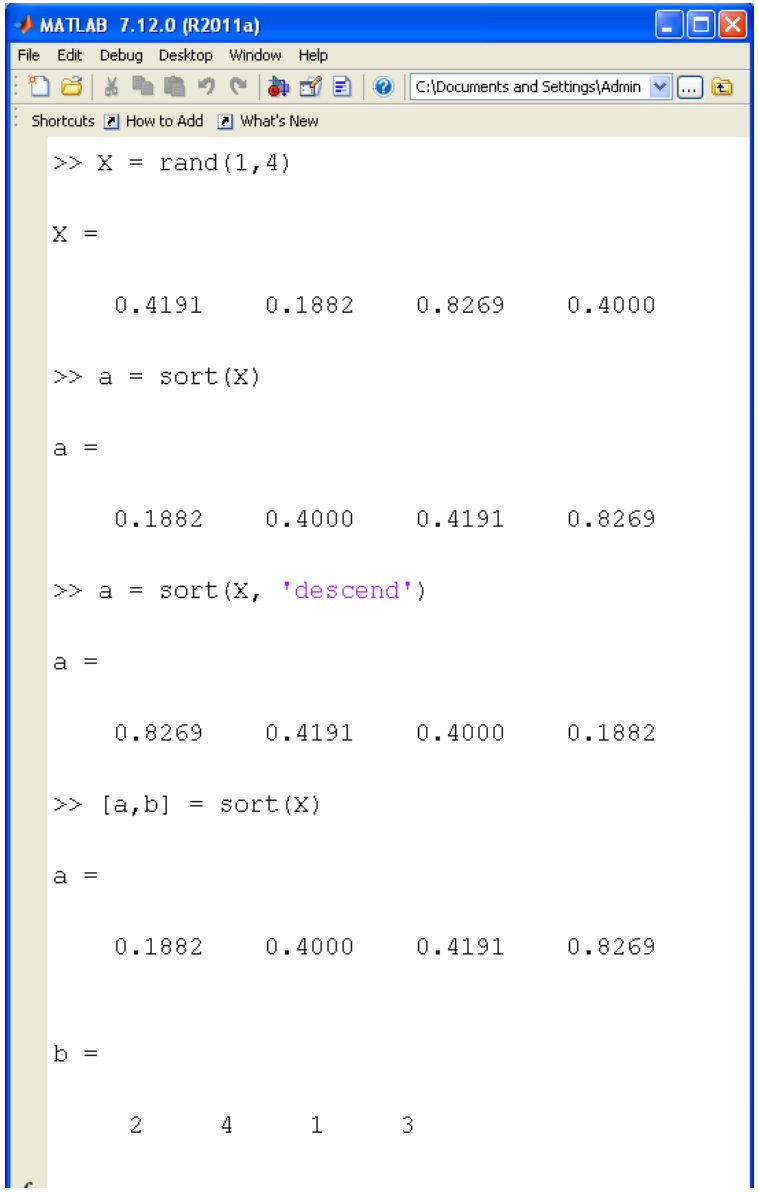
- Returns the sorted data in ascending order

a = sort(Deck, 'descend');

- Returns the sorted data in descending order

[a,b] = sort(Deck);

- a = sorted data
- b = the sort order (which element was smallest, etc.)



Sort Command & Poker

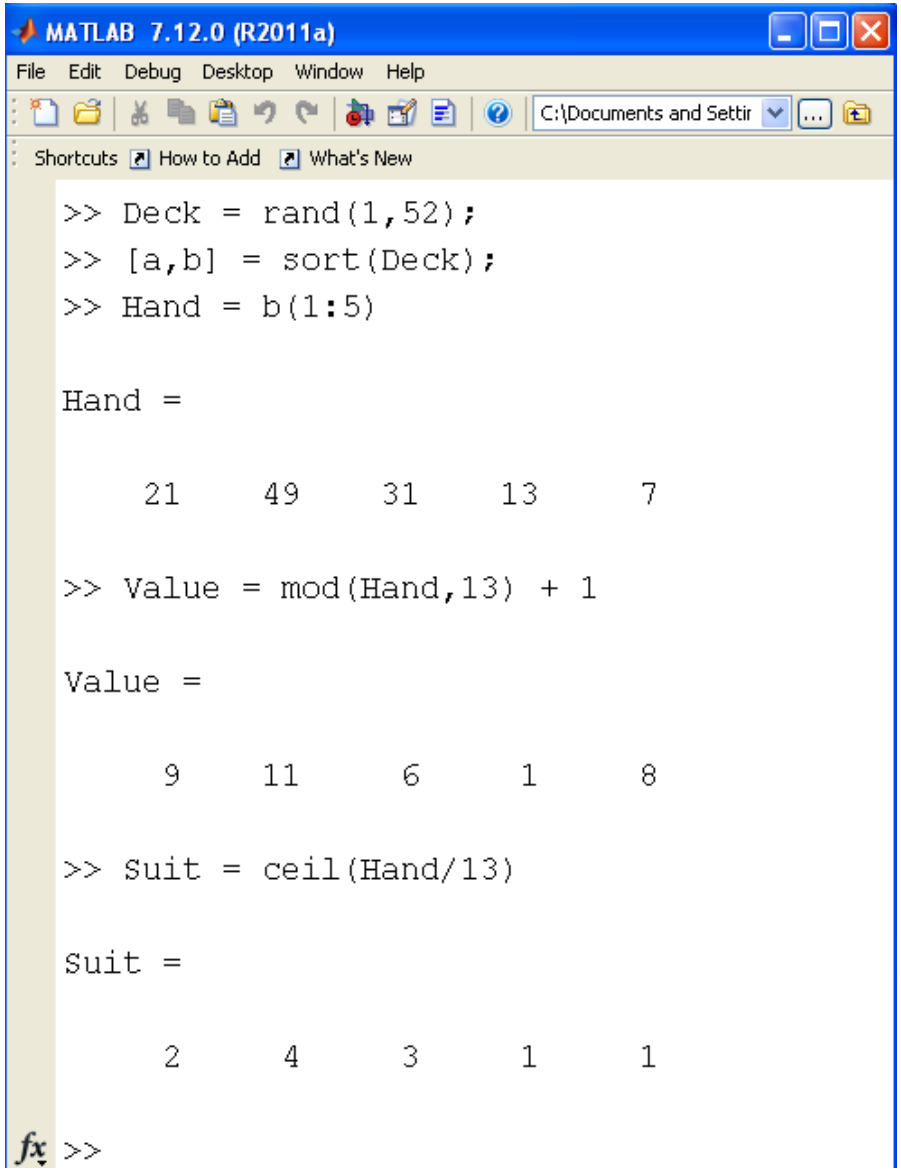
Shuffle a deck of 52 playing cards

Deal out 5 cards (poker hand)

- Hand is the card number (1..52)
- Value is Ace..King (1..13)
- Suit is Club / Diamond / Heart / Spade (1...4)

Result:

- {9d, Js, 6h, Ac, 8c}



Check for a Full-House

P(i): Count how many Aces, twos, etc. are in your hand:

Sort P to determine the frequency of pairs in descending order

- Max number of matches = 2
- Pair of 8's
- Next highest frequency = 1

Resulting hand is a pair

- If P(1) = 3 and P(2) = 2, you have a full-house

MATLAB 7.12.0 (R2011a)

File Edit Debug Desktop Window Help

C:\Documents and Settir

Shortcuts How to Add What's New

```
>> FH = 0;

P = zeros(1,13);
for i=1:13
   P(i) = sum(Value == i);
   end
P = sort(P, 'descend');

if((P(1) == 3)*(P(2) == 2))
   FH = FH + 1;
   end

disp(Value)
disp(P(1:5))
```

```
     9     8     6     1     8

     2     1     1     1     0
```

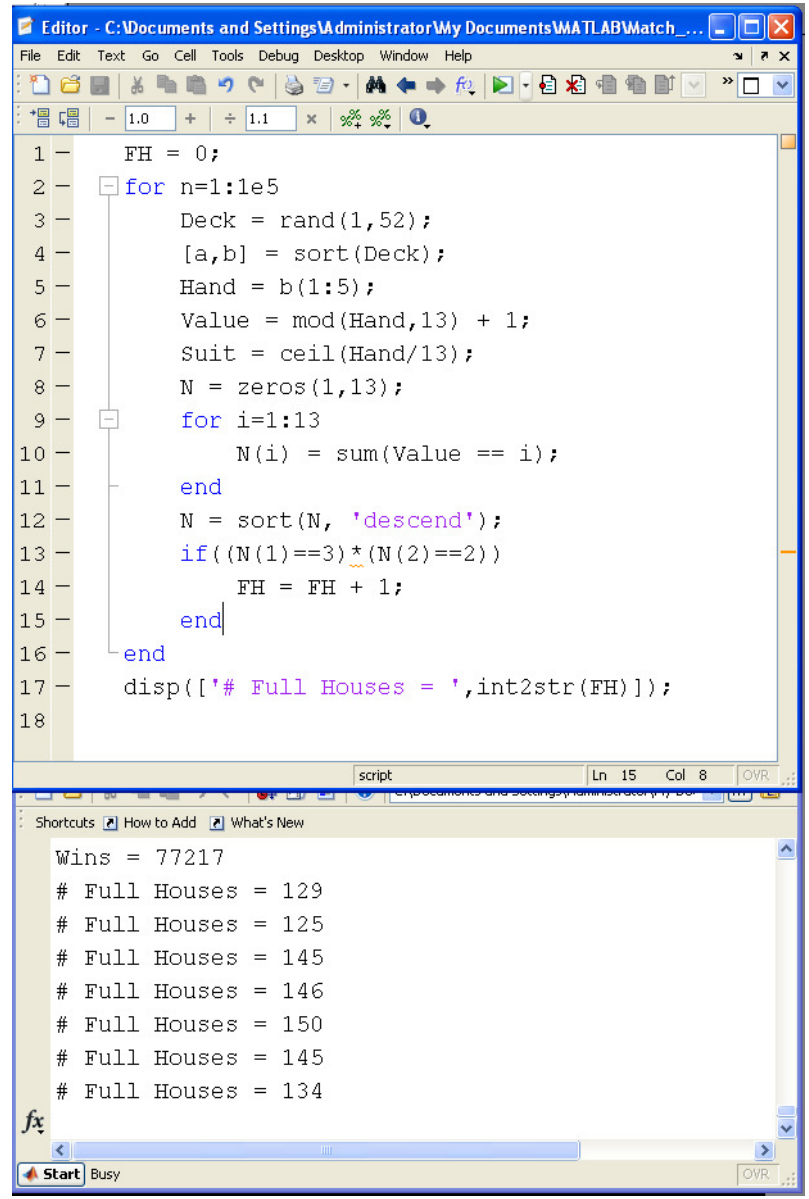
Repeat 100,000 times

- Monte-Carlo Simulation
- With 100,000 poker hands, I got a full-house between 129 and 150 times (it's random)
- Average is 139.75

The odds of getting a full-house are approximately

$$p \approx \left(\frac{139.75}{100{,}000}\right)$$

$$p \approx 715 : 1 \text{ odds against}$$



Problem:

Given these results, what are the odds of getting a full house?

- N = {129,125,145,146,150,145,134,144}

Solution:

- This is a statistics problem
- The solution is a t-Test
- This is covered in ECE 341 Random Processes & week #12 in ECE 111

Summary

Matlab is a fairly friendly computer language

You can use the command window as a calculator

- Adds, subtracts, multiplies, divides

Scripts allow you to try & modify code as you write it

For-loops let you run code multiple times

- Monte-Carlo simulations...

If-statements allow you to check for conditions

- If the sum is 25 or more...

While-loops let you run code until an event happens

- repeat until you roll a 1

Matlab Commands

Display

- format short display results to 4 decimal places
- format long display results to 13 decimal places
- format short e display using scientific notation
- format long e display using scientific notation

Polynomials

- poly(x)
- roots(x)
- conv(x,y)

Analysis

- sqrt(x) square root of x
- log(x) log base e
- log10(x) log base 10
- exp(x) e^x
- exp10(x) 10^x
- abs(x) |x|
- round(x) round to the nearest integer
- floor(x) round down (integer value of x)
- ceil(x) round up to the next integer
- real(x) real part of a complex number
- imag(x) imaginary part of a complex number
- abs(x) absolute value of x, magnitude of a complex number
- angle(x) angle of a complex number (answer in radians)
- unwrap(x) remove the discontinuity at pi (180 degrees) for a vector of angles
- sum(x) sum the columns of x
- prod(x) multiply the columns of x

Trig Functions

- sin(x) — sin(x) where x is in radians
- cos(x) — cos()
- tan(x) — tan()
- asin(x) — arcsin(x)
- acos(x) — arccos(x)
- atan(x) — arctan(x)
- atan2(y,x) — angle to a point (x,y)

Probability and Statistics

- factorial(x) — x!
- gamma(x) — x!
- rand(n,m) — create an nxm matrix of random numbers between 0 and 1
- randn(n,m) — create an nxm matrix of random numbers with a normal distribution
- length(x) — return the dimensions of x
- mean(x) — mean (average) of the columns of x
- std() — standard deviation of the columns of x

Display Functions

- plot(x) — plot x vs sample number
- plot(x,y) — plot x vs. y
- semilogx(x,y) — log(x) vs y
- semilogy(x,y) — x vs log(y)
- loglog(x,y) — log(x) vs log(y)
- mesh(x) — 3d plot where the height is the value at x(a,b)
- contour(x) — contour plot
- bar(x,y) — draw a bar graph
- xlabel('time') — label the x axis with the word 'time'
- ylabel() — label the y axis
- title() — put a title on the plot
- grid() — draw the grid lines

Useful Commands

- hold on — don't erase the current graph
- hold off — do erase the current graph

- diary — create a text file to save whatever goes to the screen
- linepace(a, b, n) — create a 1xn array starting at a, increment by b
- logspace(a,b,n) — create a 1xn array starting at 10^a going to 10^b, spaced logarithmically
- subplot() — create several plots on the same screen
- disp('hello') — display the message *hello*

Utilities

- format — set the display format
- zeros(n,m) — create an nxm matrix of zeros
- eye(n,m) — create an nxm matrix with ones on the diagonal
- ones(n,m) — create an nxm matrix of ones
- help — help using different functions
- pause(x) — pause x seconds (can be a fraction). Show the graph as well
- clock — the present time
- etime — the difference between two times
- tic — start a stopwatch
- toc — the number of seconds since tic