
Matlab Basics

ECE 111 Introduction to ECE Week #1

Please visit [Bison Academy](#) for corresponding
lecture notes, homework sets, and solutions

Becoming familiar with MATLAB

- Using the command window
- Using scripts
- Plotting with Matlab
- Random numbers in Matlab
- If-Statements
- For-Loops
- While-Loops
- Monte-Carlo Simulations



Registering with Matlab

Instructions to register with Matlab are here:

- <https://kb.ndsu.edu/page.php?id=102044>

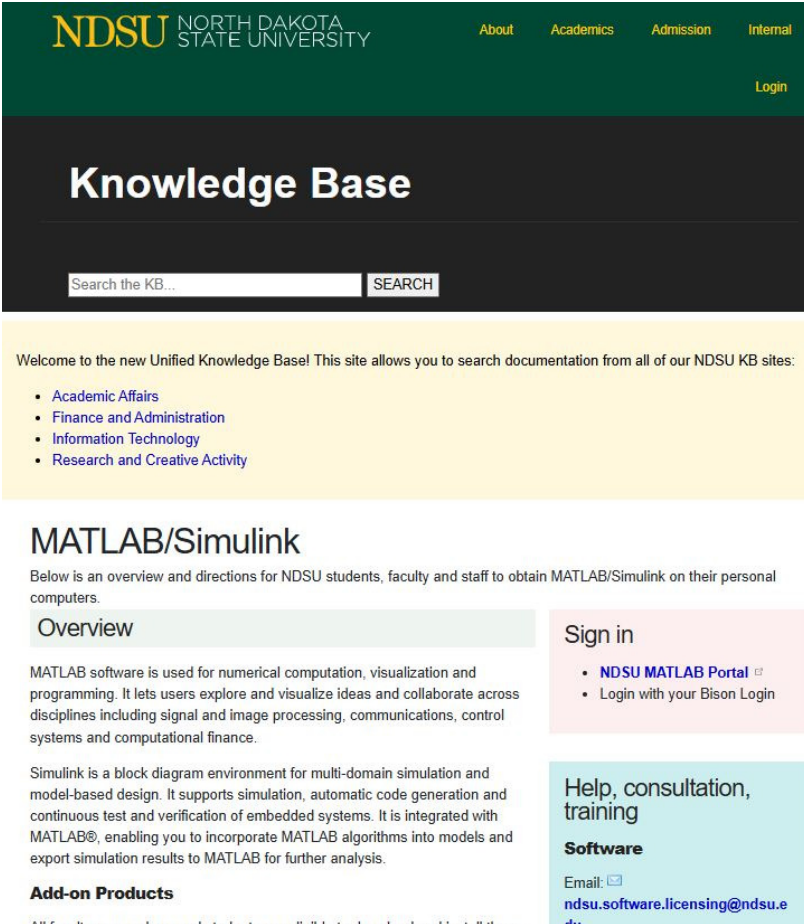
You'll need to use your NDSU email address

- xxx@ndsu.edu *this does work*
- xxx@ndus.edu *this does not work*

Toolboxes: Dozens are available. I use:

- Controls Systems (general)
- Signals and Systems (general)

You're welcome to play around with the other toolboxes. I personally don't use them.



The screenshot shows the NDSU Knowledge Base website. The header is dark green with the NDSU logo and navigation links: About, Academics, Admission, Internal, and a Login link. The main section is dark grey with the title 'Knowledge Base' and a search bar. Below the search bar, a yellow box contains a welcome message and a list of categories: Academic Affairs, Finance and Administration, Information Technology, and Research and Creative Activity. The 'MATLAB/Simulink' section follows, providing an overview of the software and its use. To the right of the overview, there are two sidebars: 'Sign in' with links to the NDSU MATLAB Portal and login instructions, and 'Help, consultation, training' with a software section and an email contact.

NDSU NORTH DAKOTA STATE UNIVERSITY

About Academics Admission Internal Login

Knowledge Base

Search the KB... SEARCH

Welcome to the new Unified Knowledge Base! This site allows you to search documentation from all of our NDSU KB sites:

- Academic Affairs
- Finance and Administration
- Information Technology
- Research and Creative Activity

MATLAB/Simulink

Below is an overview and directions for NDSU students, faculty and staff to obtain MATLAB/Simulink on their personal computers.

Overview

MATLAB software is used for numerical computation, visualization and programming. It lets users explore and visualize ideas and collaborate across disciplines including signal and image processing, communications, control systems and computational finance.

Simulink is a block diagram environment for multi-domain simulation and model-based design. It supports simulation, automatic code generation and continuous test and verification of embedded systems. It is integrated with MATLAB®, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.

Add-on Products

All faculty, researchers and students are eligible to download and install these

Sign in

- [NDSU MATLAB Portal](#)
- Login with your Bison Login

Help, consultation, training

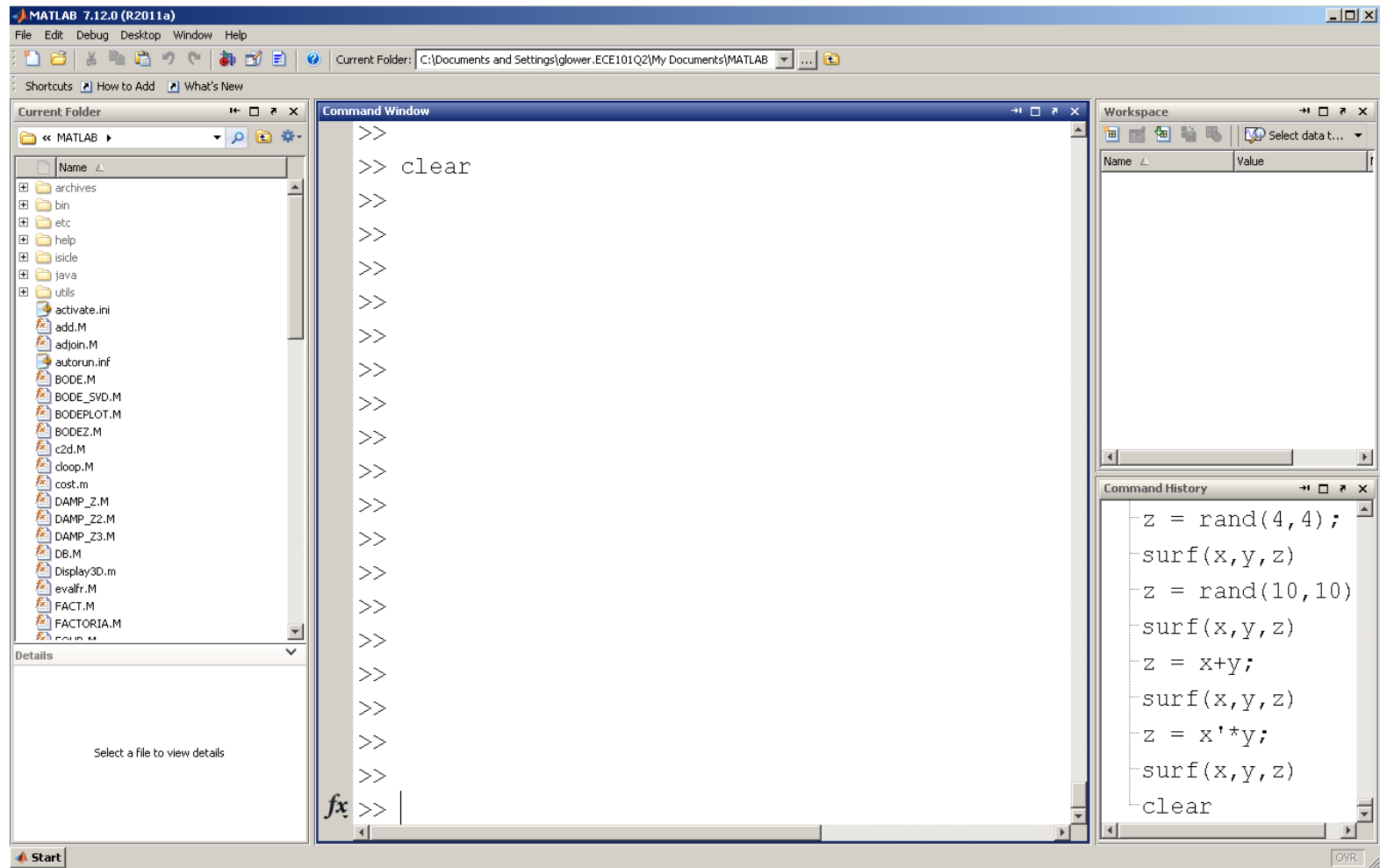
Software

Email: ndsu.software.licensing@ndsu.edu

General environment and the console

Startup Screen:

- I usually close everything down except the command window



Command Window

Matlab works like a calculator

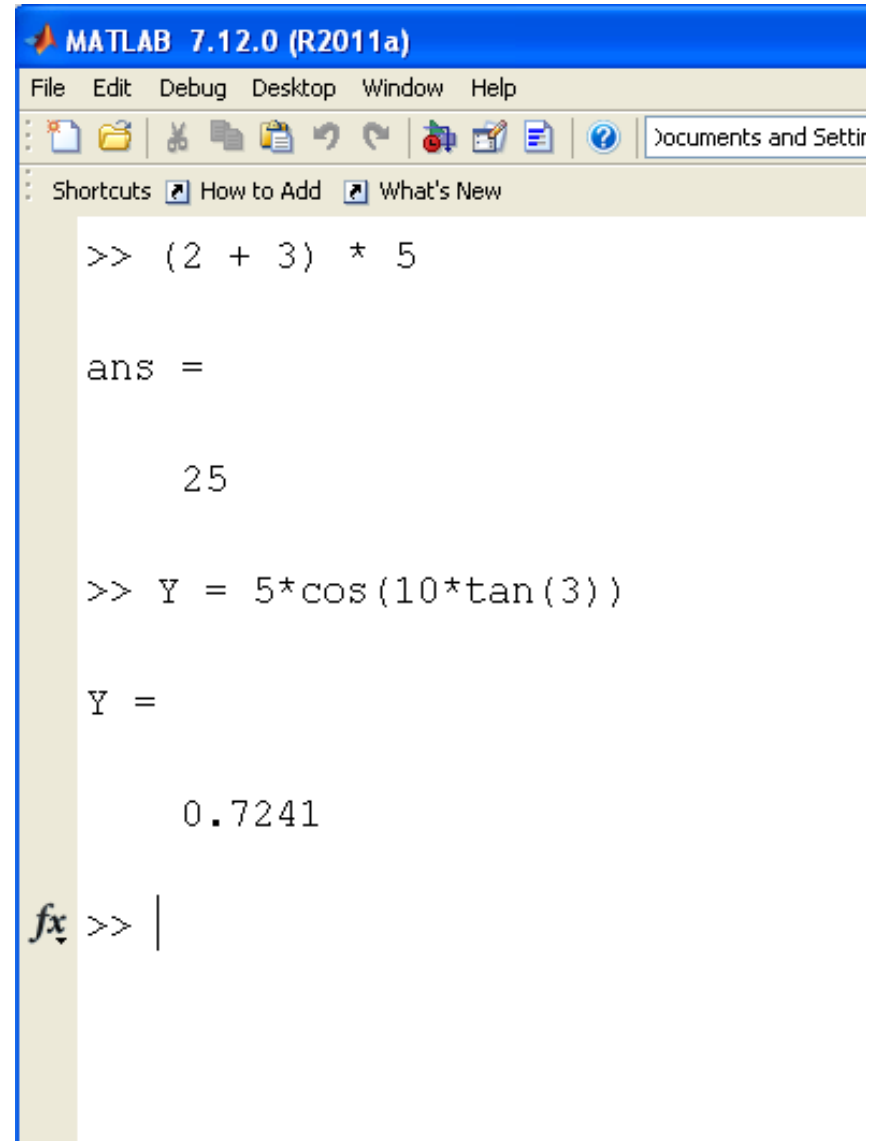
- Solve

$(2 + 3) * 5$

- Solve

$Y = 5 * \cos(10 * \tan(3))$

You type it the way it looks:

A screenshot of the MATLAB 7.12.0 (R2011a) Command Window. The window has a blue title bar with the MATLAB logo and version information. Below the title bar is a menu bar with 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. Underneath the menu bar is a toolbar with various icons for file operations, editing, and help. Below the toolbar is a status bar with 'Shortcuts', 'How to Add', and 'What's New'. The main area of the window is a light beige background with a vertical blue line on the left. It shows the command prompt '>>' followed by the expression '(2 + 3) * 5'. The result 'ans = 25' is displayed below. Then, the command prompt '>>' is followed by the expression 'Y = 5*cos(10*tan(3))'. The result 'Y = 0.7241' is displayed below. At the bottom, the command prompt '>>' is followed by a vertical bar '|'.

```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
[Icons]
Shortcuts How to Add What's New

>> (2 + 3) * 5

ans =

    25

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

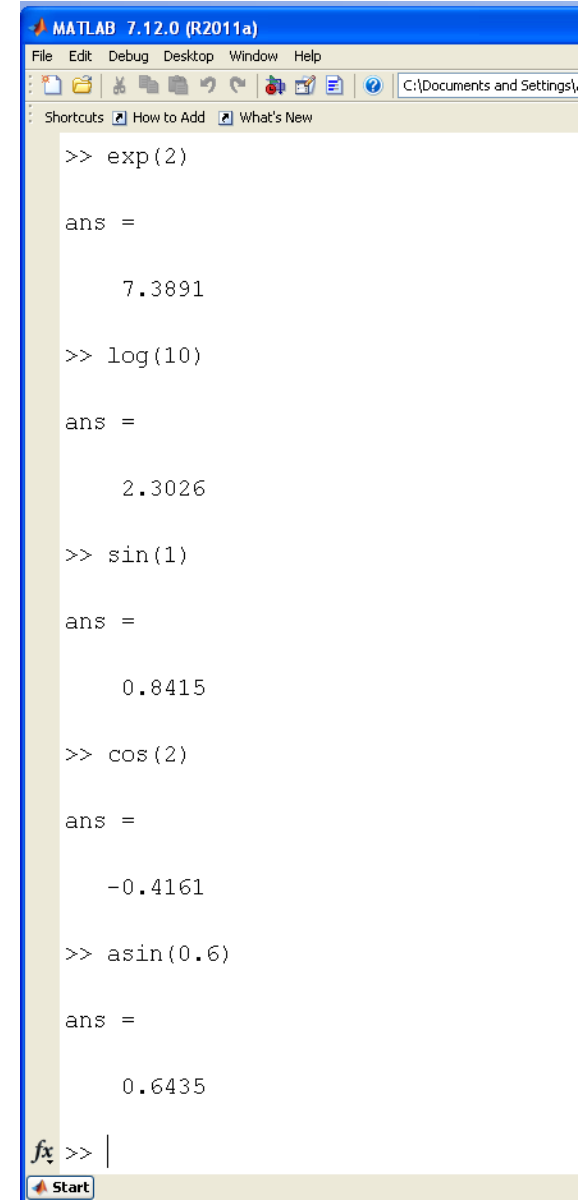
Y =

    0.7241

fx >> |
```

Matlab Function Names

- pi π
- exp(x) e^x
- 10^x 10^x
- log(x) $\ln(x)$
- log10(x) $\log_{10}(x)$
- sin(x) $\sin(x)$ *units = radians*
- cos(x) $\cos(x)$ " "
- tan(x) $\tan(x)$ " "
- asin(x) $\arcsin(x)$
- acos(x) $\arccos(x)$
- atan(y,x) $\arctan(y/x)$
- + many more



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Documents and Settings\...
Shortcuts How to Add What's New

>> exp(2)

ans =

    7.3891

>> log(10)

ans =

    2.3026

>> sin(1)

ans =

    0.8415

>> cos(2)

ans =

   -0.4161

>> asin(0.6)

ans =

    0.6435

fx >> |
Start
```

Order of Operations

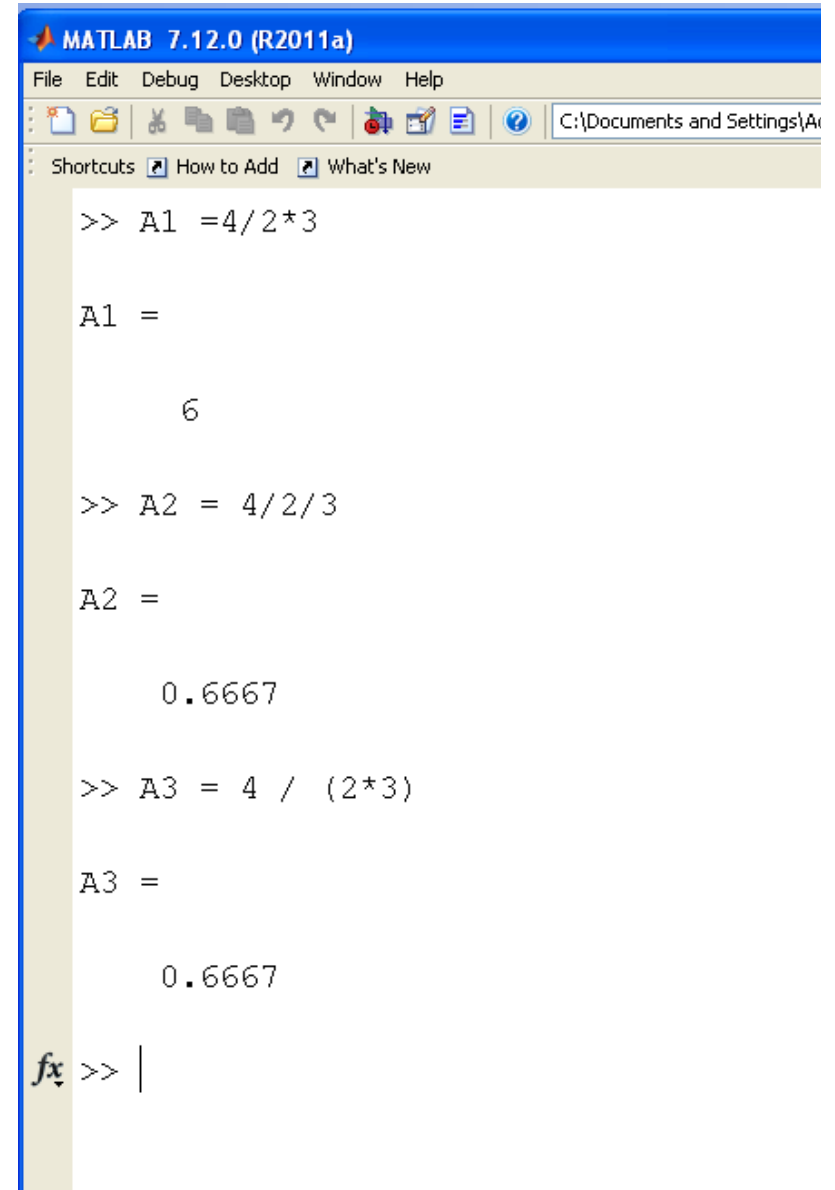
- 1st: $\wedge$
- 2nd: $*$, $/$
- 3rd: $+$, $-$

Equations are executed

- By order of operations, then
- Left to right

Paranthesis never hurt,

- They can avoid confusion



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Documents and Settings\A...
Shortcuts How to Add What's New

>> A1 = 4/2*3

A1 =

        6

>> A2 = 4/2/3

A2 =

    0.6667

>> A3 = 4 / (2*3)

A3 =

    0.6667

fx >> |
```

Matlab as a Calculator

You can define variables as you go

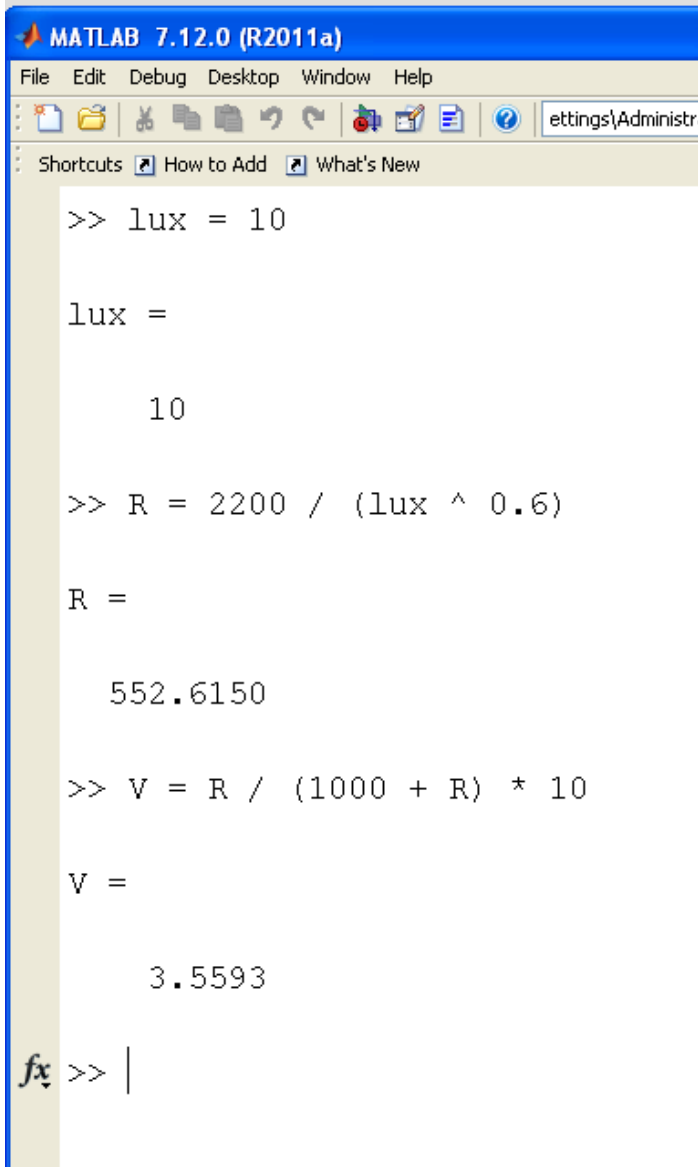
- 1st character must be a letter
- 2nd onward can be letters or numbers
- Case sensitive

Example: Light Sensor

$$R = \left(\frac{2200}{(lux)^{0.6}} \right) \Omega$$

$$V = \left(\frac{R}{1000+R} \right) 10V$$

Find R and V @ 10 Lux



The image shows the MATLAB 7.12.0 (R2011a) interface. The Command Window displays the following commands and results:

```
>> lux = 10

lux =

    10

>> R = 2200 / (lux ^ 0.6)

R =

    552.6150

>> V = R / (1000 + R) * 10

V =

    3.5593

fx >> |
```


Doing Several Operations at Once

Matlab is a matrix language

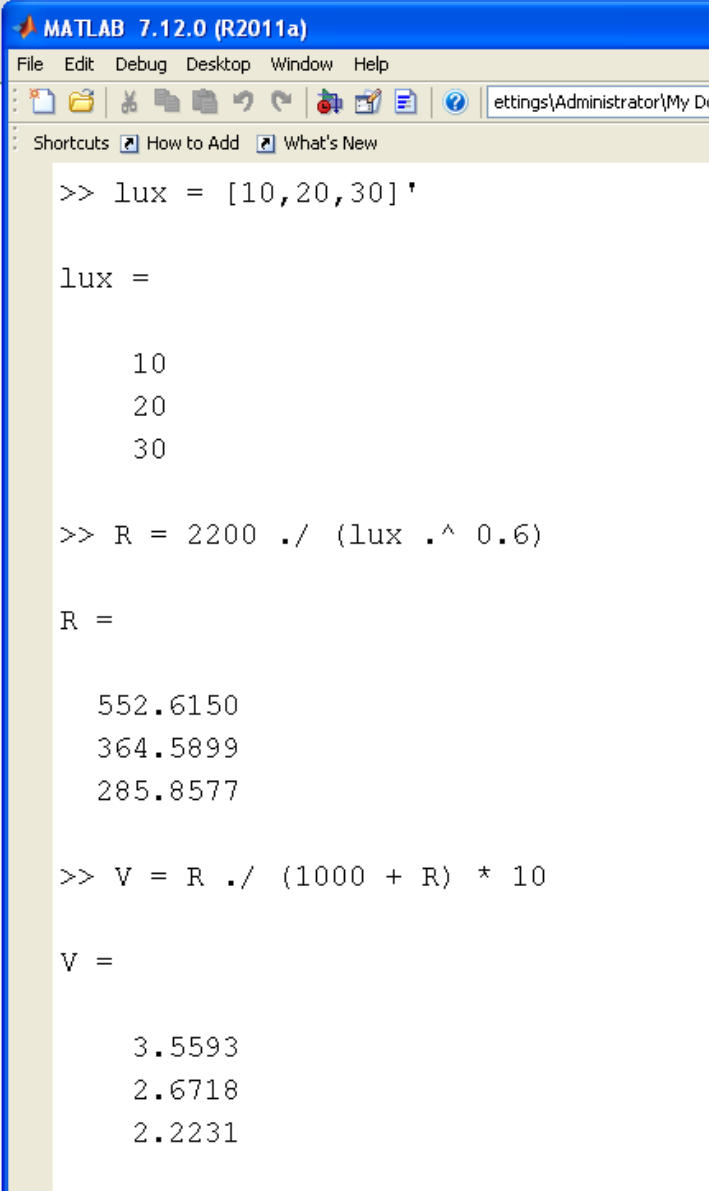
```
[      start of a matrix
,      next column (a space also works)
;      next row
]      end of matrix.
'      transpose
```

Multiplication, Power

- *, ^
- Matrix operations (coming soon)

Dot-Notation

- .* ./ , ^
- Element-by-element operations
- Allows you to do several operations at once

A screenshot of the MATLAB 7.12.0 (R2011a) software interface. The window title is "MATLAB 7.12.0 (R2011a)". The menu bar includes "File", "Edit", "Debug", "Desktop", "Window", and "Help". The toolbar contains icons for file operations (new, open, save, print, etc.) and a search icon. The command window shows the following code and output:

```
>> lux = [10,20,30]';

lux =

    10
    20
    30

>> R = 2200 ./ (lux .^ 0.6)

R =

    552.6150
    364.5899
    285.8577

>> V = R ./ (1000 + R) * 10

V =

    3.5593
    2.6718
    2.2231
```

Formatting Output

- Terminate a line with a semi-colon if you don't want the result displayed
- Leave off the semi-colon if you *do* want to see the result

```
format short  
pi
```

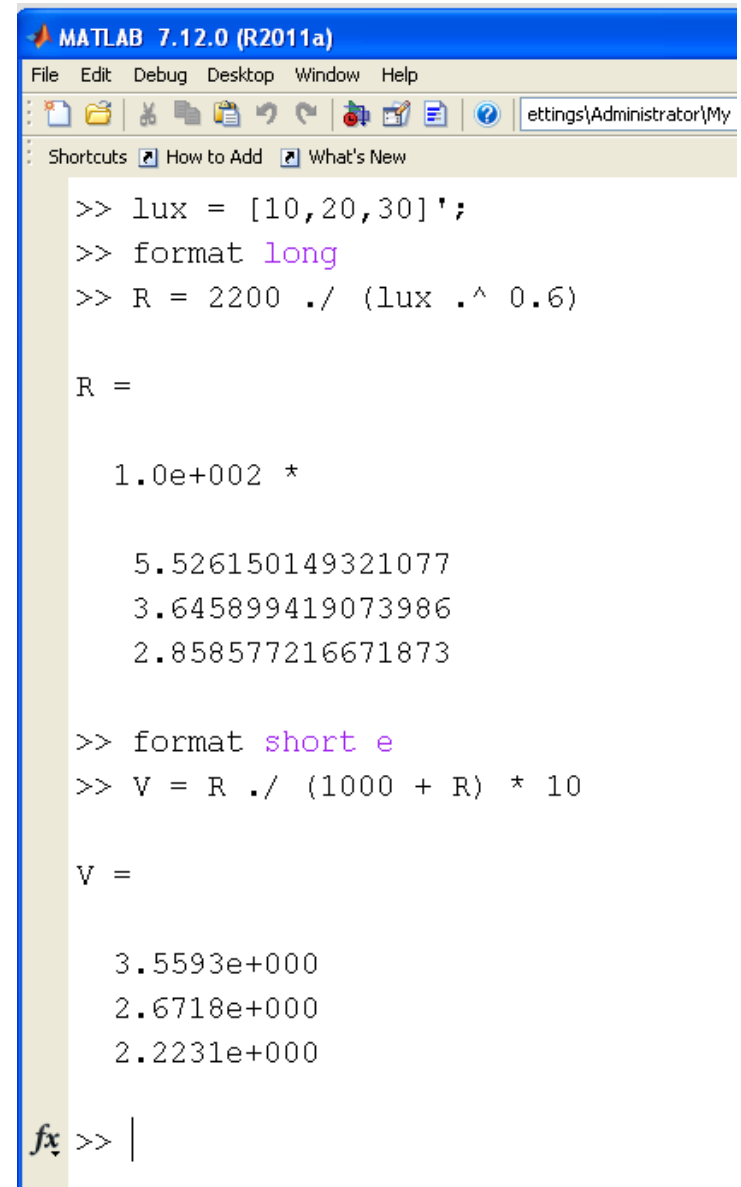
```
3.1416
```

```
format long  
pi
```

```
3.141592653589793
```

```
format shorteng  
pi^30
```

```
821.2893e+012
```

A screenshot of the MATLAB 7.12.0 (R2011a) command window. The window has a blue title bar and a menu bar with File, Edit, Debug, Desktop, Window, and Help. Below the menu bar is a toolbar with various icons. The command window shows the following code and output:

```
>> lux = [10,20,30]';  
>> format long  
>> R = 2200 ./ (lux.^ 0.6)  
  
R =  
  
1.0e+002 *  
  
5.526150149321077  
3.645899419073986  
2.858577216671873  
  
>> format short e  
>> V = R ./ (1000 + R) * 10  
  
V =  
  
3.5593e+000  
2.6718e+000  
2.2231e+000  
  
fx >> |
```

Matlab as a Graphing Calculator

Matlab has pretty good graphics

This is useful if you want to know what happens over a range of values.

Example: Find R for $1 < \text{lux} < 100$

$$R = \left(\frac{2200}{(\text{lux})^{0.6}} \right) \Omega$$

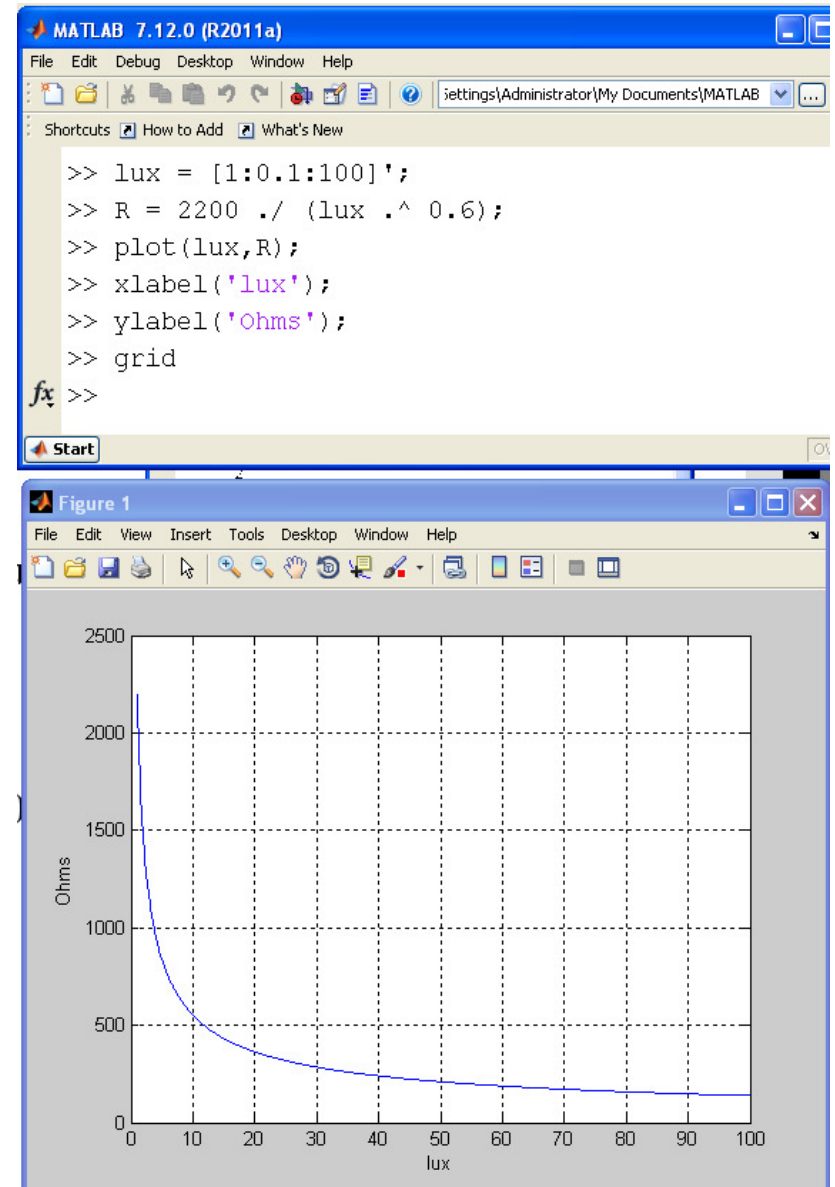
Linear spacing

- 1 to 100 lux, step size = 0.1

```
lux = [1 : 0.1 : 100]';
```

Log spacing from 10^{-2} to 10^3 with 100 points

```
lux = logspace(-2, 3, 100)';
```



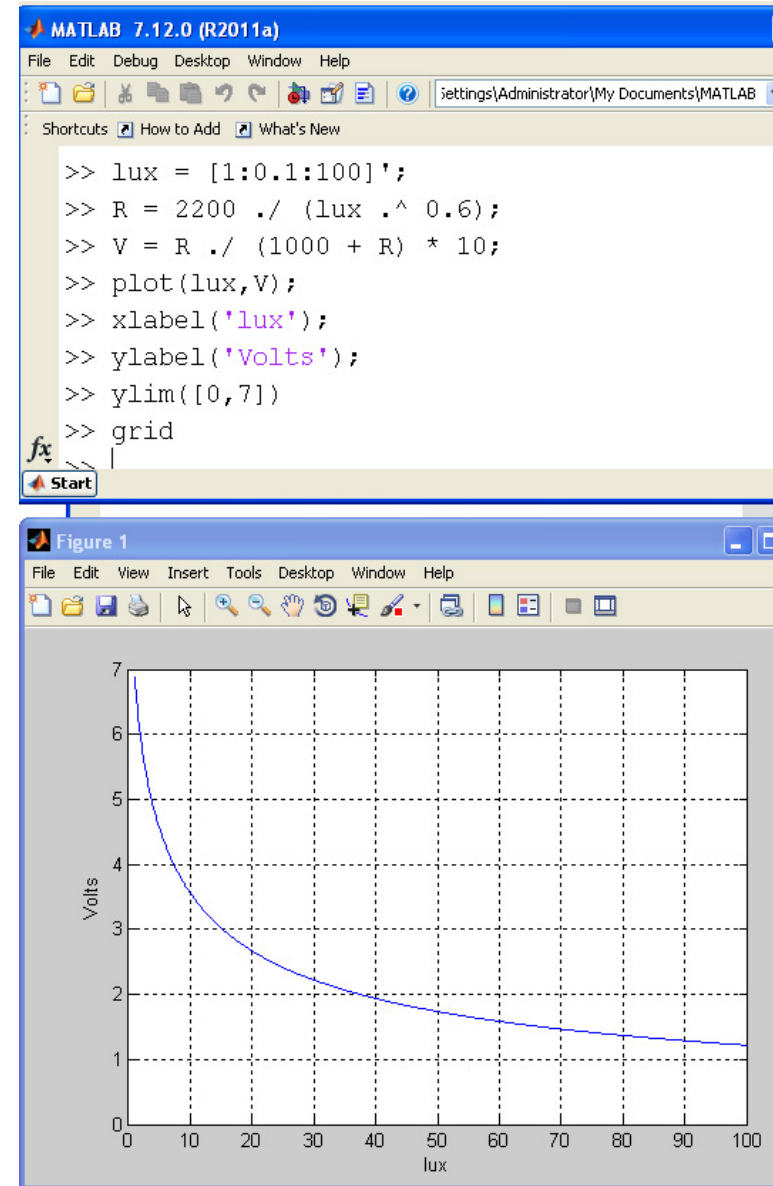
Matlab as a Graphing Calculator

Graphs are useful

- They show you how the two variables are related
- They allow you to determine V over a range of lux
- They allow you to determine lux if you know V

Example: If you read 2.00 Volts, what is the light level?

- Read it off the graph
- light = 36 lux (approx)

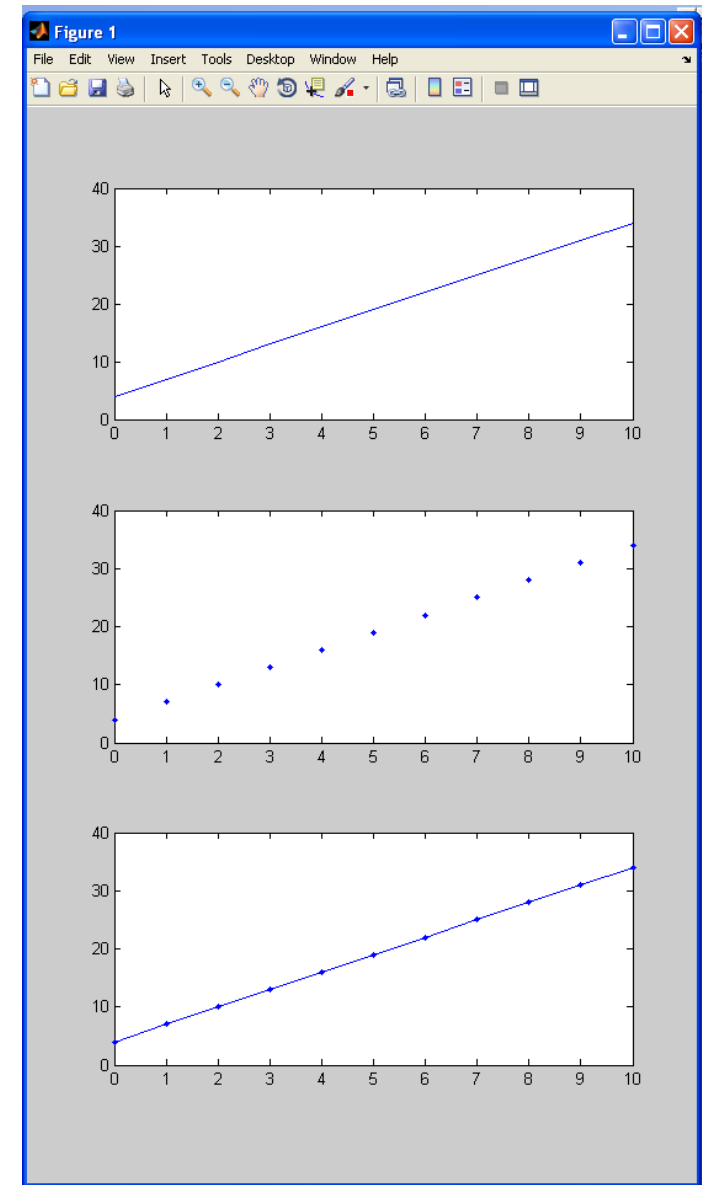


Plotting Functions in Matlab:

Matlab has some pretty good graphics capabilities.

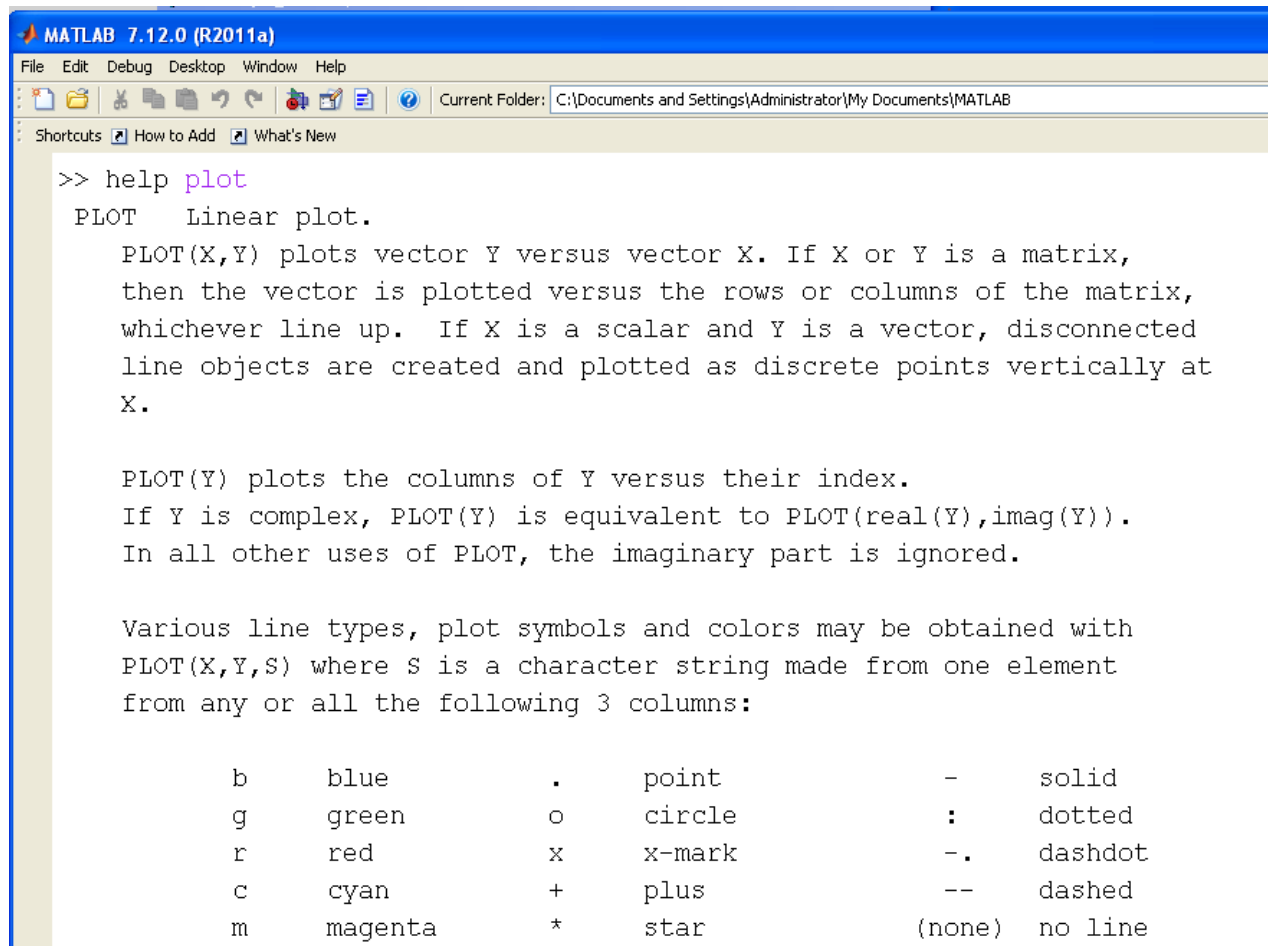
Matlab Plot Command	x axis	y axis	type of function
<code>plot(x,y)</code>	linear	linear	$y = ax + b$
<code>semilogx(x,y)</code>	log()	linear	$y = a \log(bx)$
<code>semilogy(x,y)</code>	linear	log()	$y = a e^{bx}$
<code>loglog(x,y)</code>	log()	log()	$y = a \cdot b^x$
<code>subplot(abc)</code>	Create '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, '.-');
```



Matlab Help

If you forget how to use a function, type help



The image shows a screenshot of the MATLAB 7.12.0 (R2011a) software interface. The title bar at the top reads "MATLAB 7.12.0 (R2011a)". Below it is a menu bar with "File", "Edit", "Debug", "Desktop", "Window", and "Help". A toolbar with various icons is visible. The "Current Folder" path is shown as "C:\Documents and Settings\Administrator\My Documents\MATLAB". The main workspace area displays the command prompt with the input ">> help plot". The output shows the help text for the PLOT function, including its syntax, usage, and a table of line styles and markers.

```
>> help plot
PLOT    Linear plot.
        PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
        then the vector is plotted versus the rows or columns of the matrix,
        whichever line up. If X is a scalar and Y is a vector, disconnected
        line objects are created and plotted as discrete points vertically at
        X.

        PLOT(Y) plots the columns of Y versus their index.
        If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
        In all other uses of PLOT, the imaginary part is ignored.

        Various line types, plot symbols and colors may be obtained with
        PLOT(X,Y,S) where S is a character string made from one element
        from any or all the following 3 columns:
```

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line

Multiple Plots on the same graph:

```
plot(x1,y1, x2,y2, x3,y3)  
plot(x, [y1,y2,y3])
```

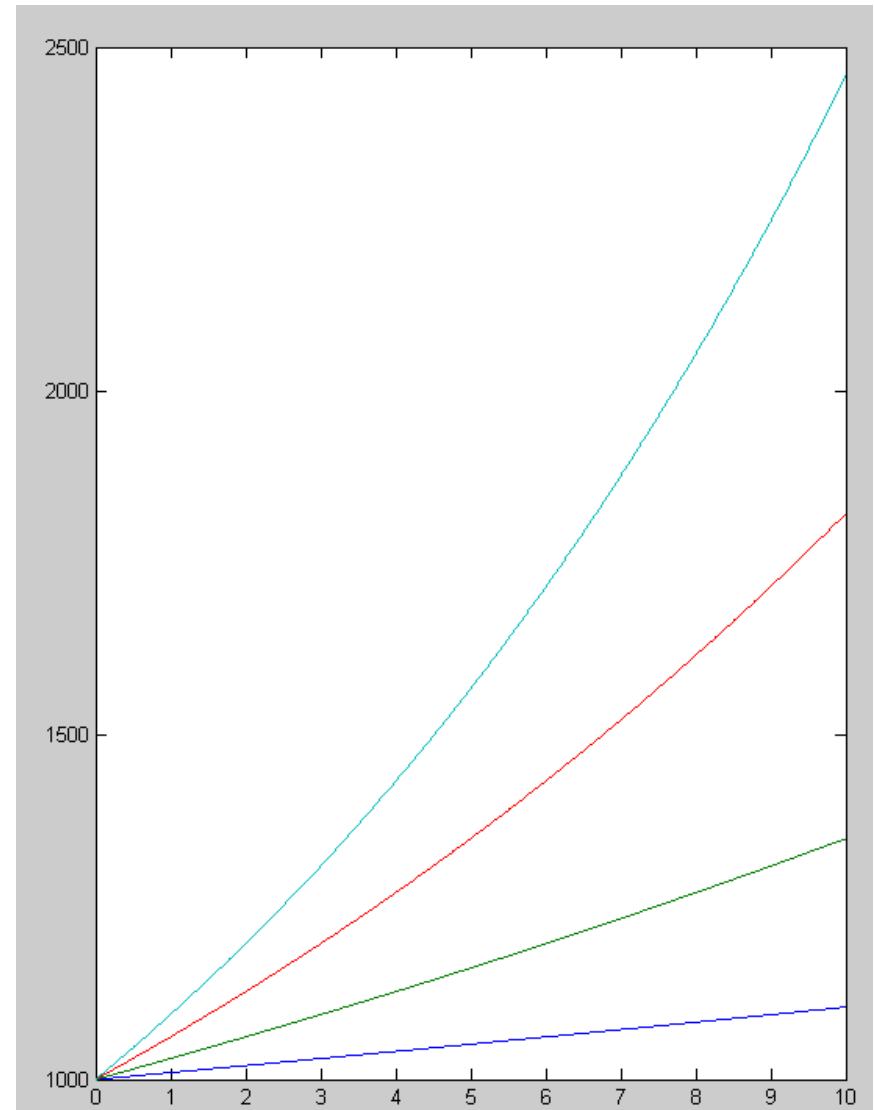
Invest \$1000 for 10 years at...

- 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

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

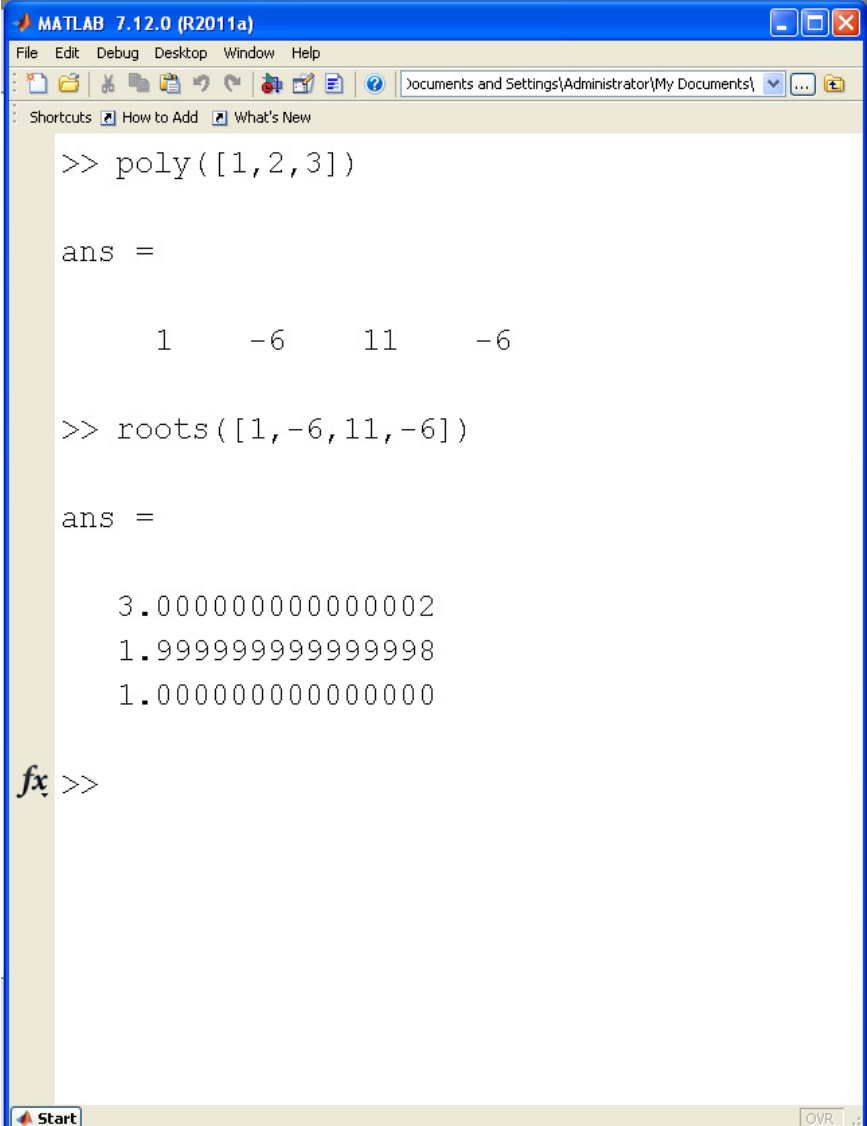
`poly([1,2,3])`

1 -6 11 -6

$$\begin{aligned} y &= x^3 - 6x^2 + 11x - 6 \\ &= (x-1)(x-2)(x-3) \end{aligned}$$

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

3.0000
2.0000
1.0000



The image shows a MATLAB 7.12.0 (R2011a) command window. The title bar is blue with the MATLAB logo and version information. The menu bar includes File, Edit, Debug, Desktop, Window, and Help. The toolbar contains icons for file operations and debugging. The command window shows the following commands and outputs:

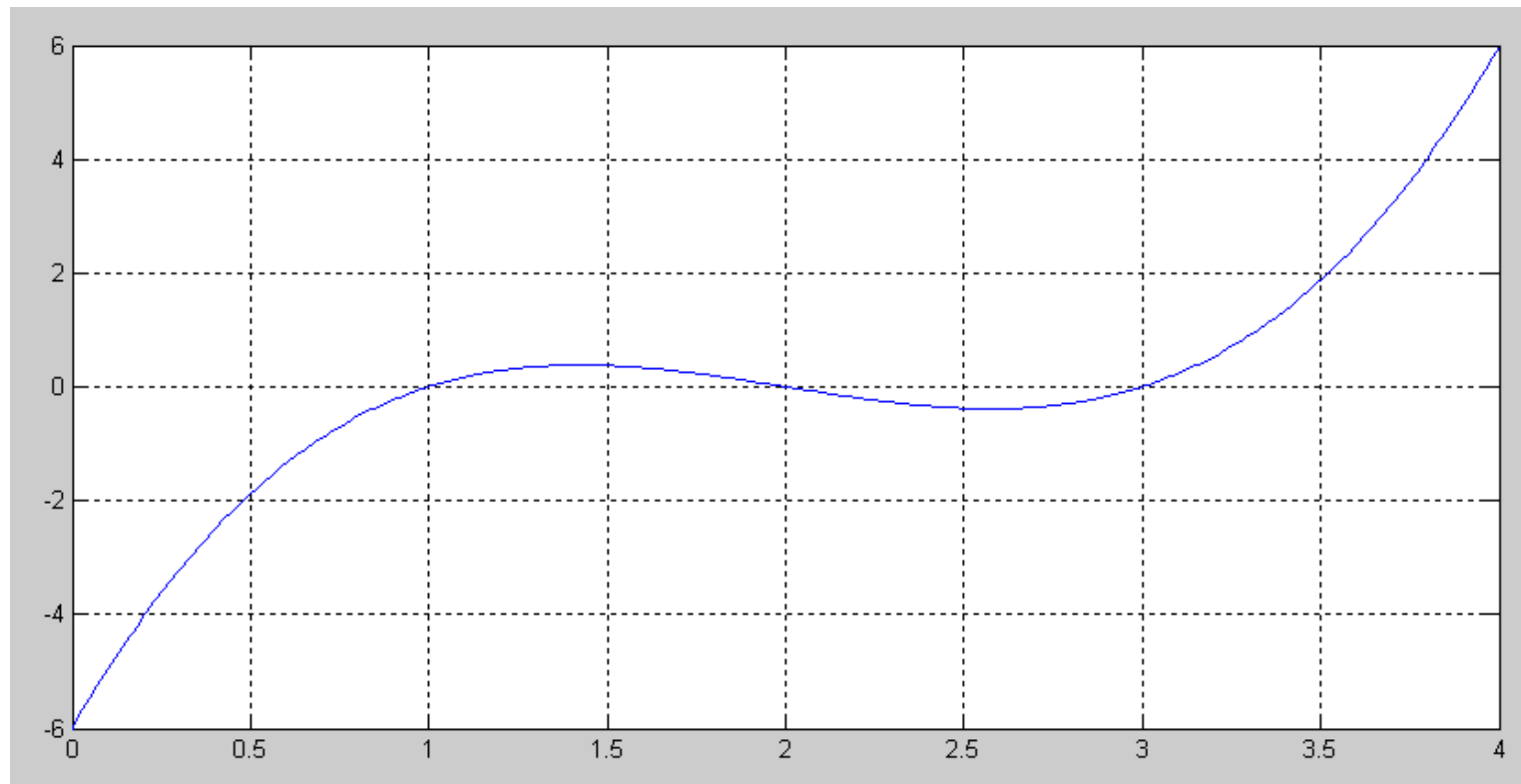
```
>> poly([1,2,3])  
  
ans =  
  
     1     -6     11     -6  
  
>> roots([1,-6,11,-6])  
  
ans =  
  
    3.000000000000000  
    1.999999999999999  
    1.000000000000000  
  
fx >>
```

The bottom status bar shows the Start button and the OVR (Overwrite) indicator.

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
```



Change the Problem to Fit the Solution

roots() finds the zero crossings of a polynomial

$$0 = x^3 + 5x^2 + 7x + 2$$

If you want to find a different answer, change the problem

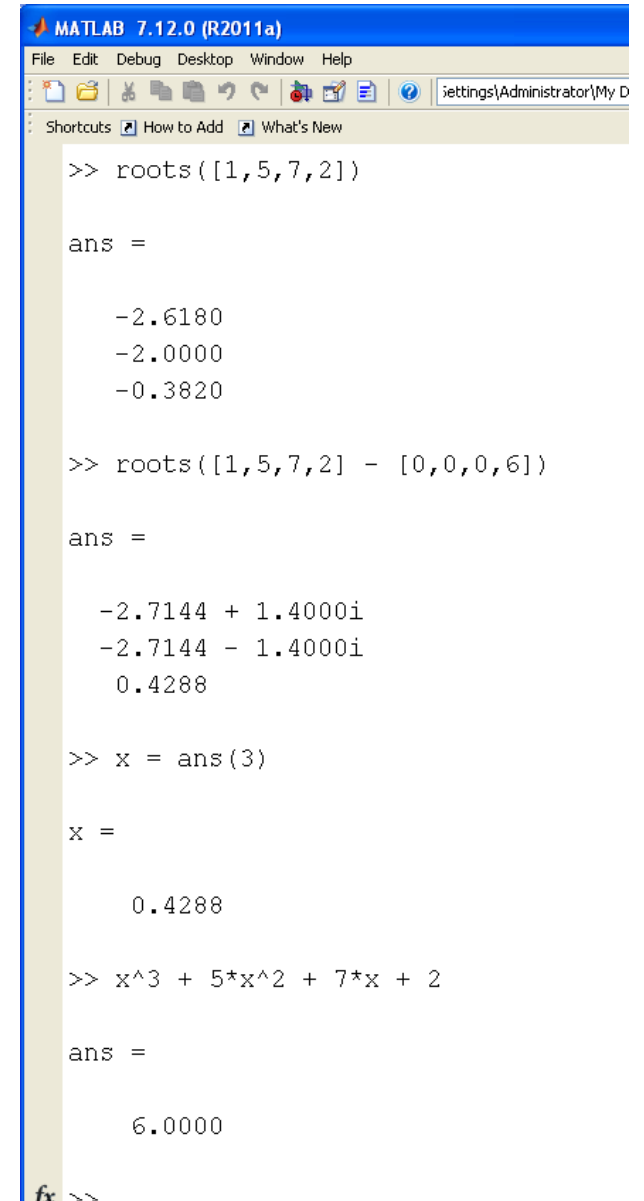
$$6 = x^3 + 5x^2 + 7x + 2$$

becomes

$$0 = (x^3 + 5x^2 + 7x + 2) - (6)$$

Note that

- $6 = 0x^3 + 0x^2 + 0x + 6$
- You have to use matrices with similar dimensions



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
Shortcuts How to Add What's New

>> roots([1,5,7,2])

ans =

    -2.6180
    -2.0000
    -0.3820

>> roots([1,5,7,2] - [0,0,0,6])

ans =

    -2.7144 + 1.4000i
    -2.7144 - 1.4000i
     0.4288

>> x = ans(3)

x =

     0.4288

>> x^3 + 5*x^2 + 7*x + 2

ans =

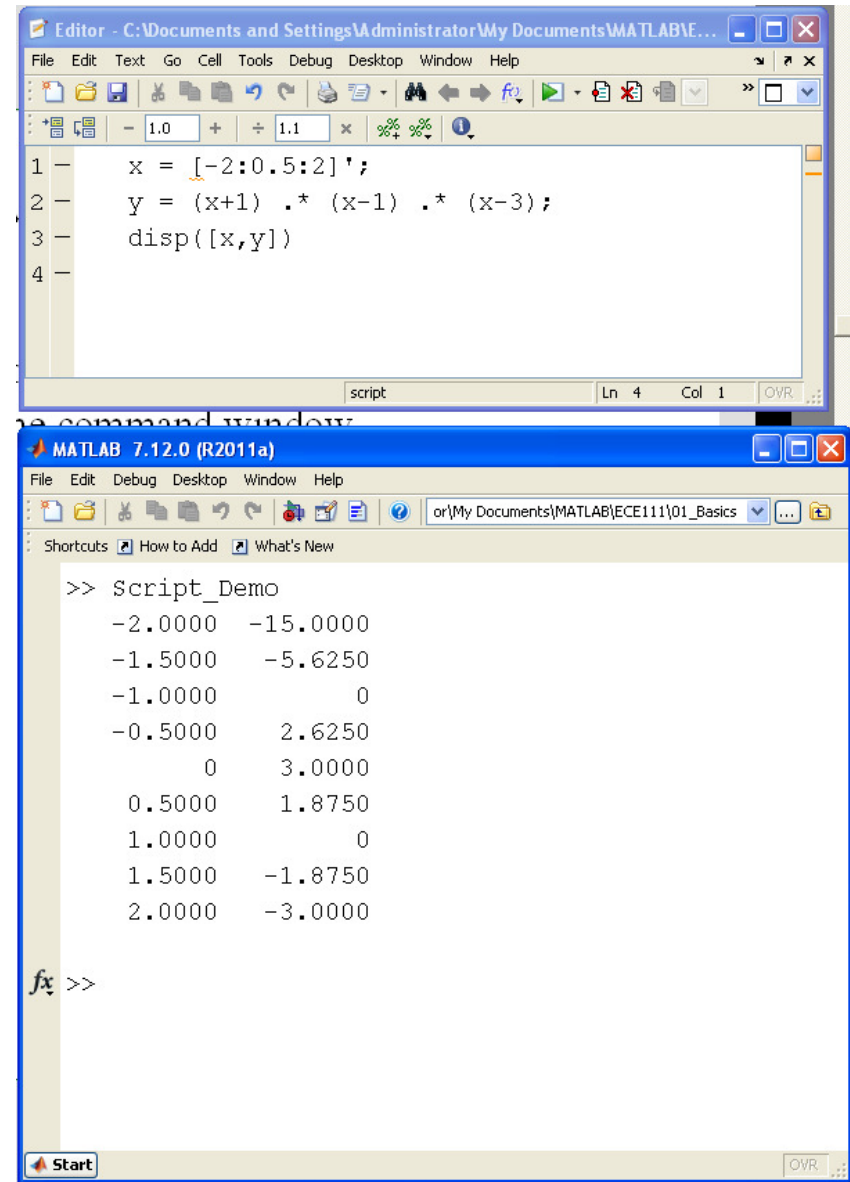
     6.0000

fx >>
```

Matlab Scripts

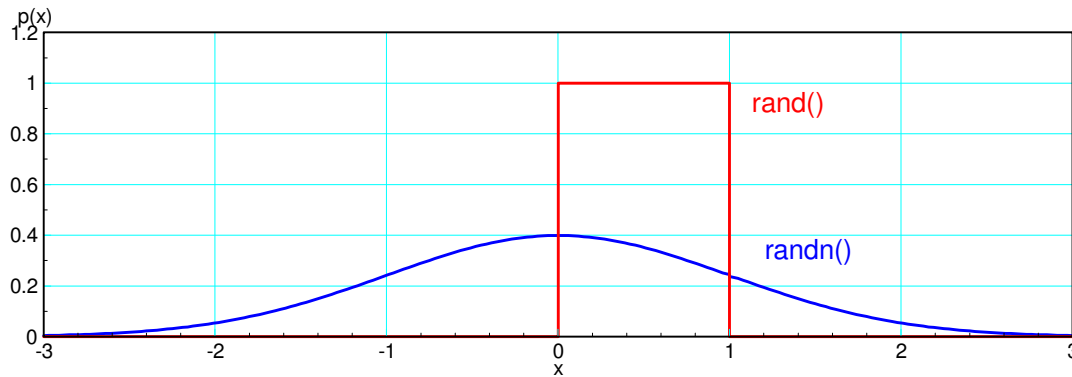
Instead of typing the same set of commands over-and-over again, you can place these Matlab commands in a file (a Matlab script)

- The file must have a .m extension
- You can execute this script using the green arrow
- You can execute this script by calling it from the command window



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)

```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
>> rand

ans =

    0.6463

>> randn(1,3)

ans =

    1.0933    1.1093   -0.8637

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

ans =

     6     6     2

fx >> |
```

If-Statement

if - end
if - else - end
if - elseif - end

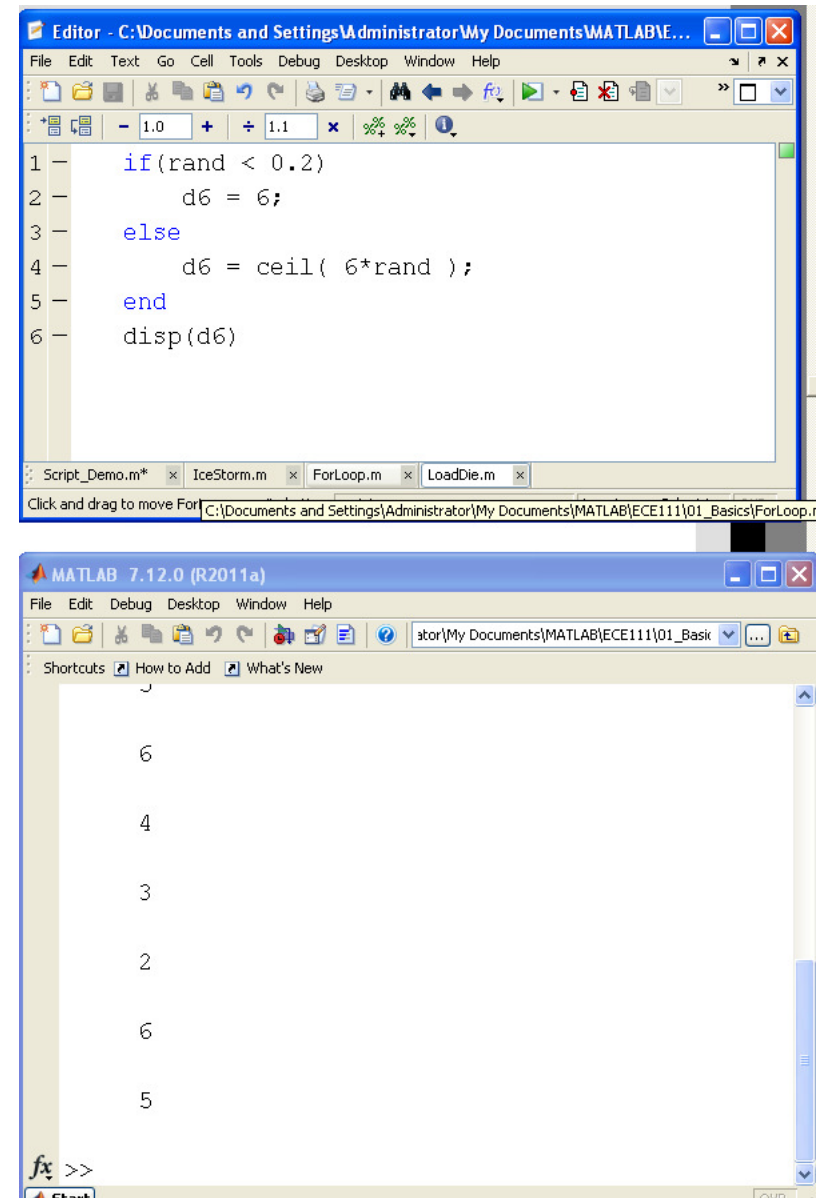
Do a set of operations if a statement is true

Valid boolean statements:

$(N == 3)$	$N = 3$
$(N > 3)$	$N > 3$
$(N \geq 3)$	$N \geq 3$
$(N \neq 3)$	$N \neq 3$
$(N \geq 3) * (N \leq 7)$	$3 \leq N \leq 7$

Example: Roll a loaded die

- 20% of the time you always roll a 6
- The rest of the time it's a fair die
- Each time you run the script, you get a new die roll



For-Loops

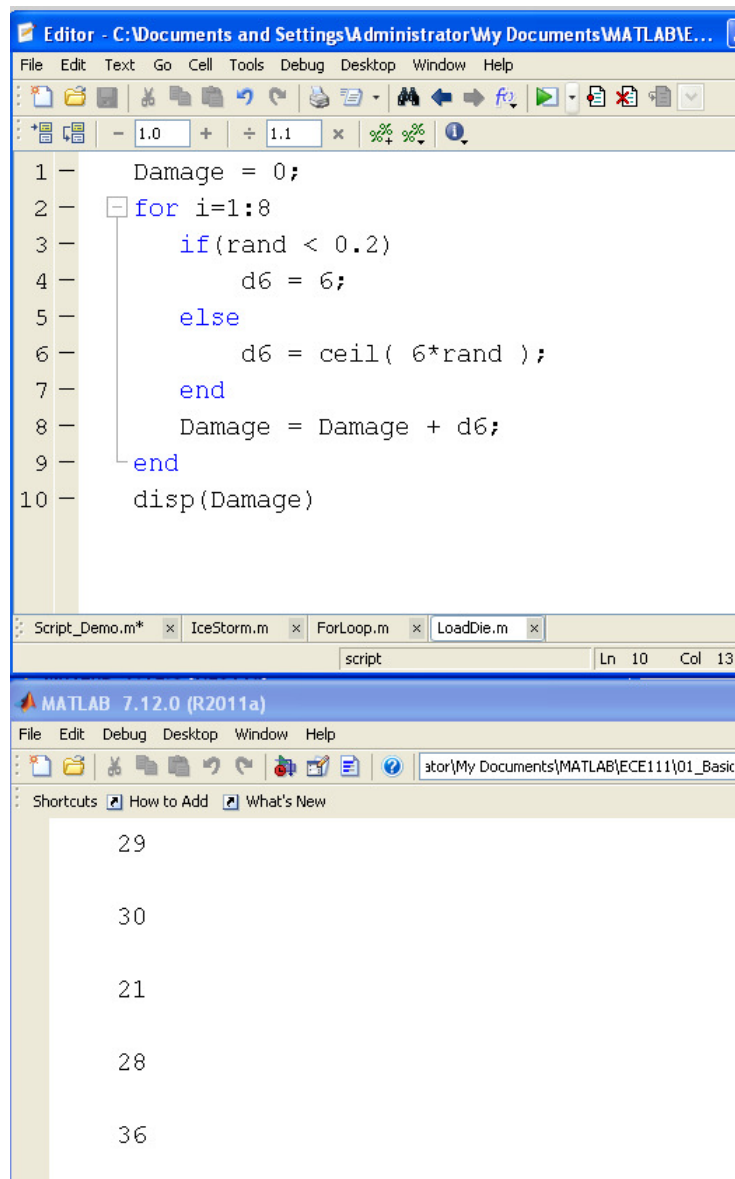
```
for i=1:100
    Matlab commands
end
```

Repeat a set of commands a fixed number of times

Terminate with an *end* statement

Example: Cast a level-8 Fireball

- $y = 8d6$
- Use loaded dice (20% chance of a 6)
- Each time you run the script, you get a different result



The image shows a MATLAB 7.12.0 (R2011a) environment. The Editor window displays a script with the following code:

```
1 Damage = 0;
2 for i=1:8
3     if(rand < 0.2)
4         d6 = 6;
5     else
6         d6 = ceil( 6*rand );
7     end
8     Damage = Damage + d6;
9 end
10 disp(Damage)
```

The Command Window shows the output of the script, which is a single value: 29. Below this, the Command Window shows the results of running the script multiple times, displaying a list of values: 29, 30, 21, 28, and 36.

Monte-Carlo Simulations

- One extremely useful capability of Matlab is to run Monte Carlo simulations
- To find the probability of an event, repeat an experiment 100,000 times
- The probability is then roughly the percentage of the time the outcome happened

Procedure:

- Write a script to run an experiment one time
 - Once that works, repeat 100,000 times
 - (place the code inside a for-loop)
-

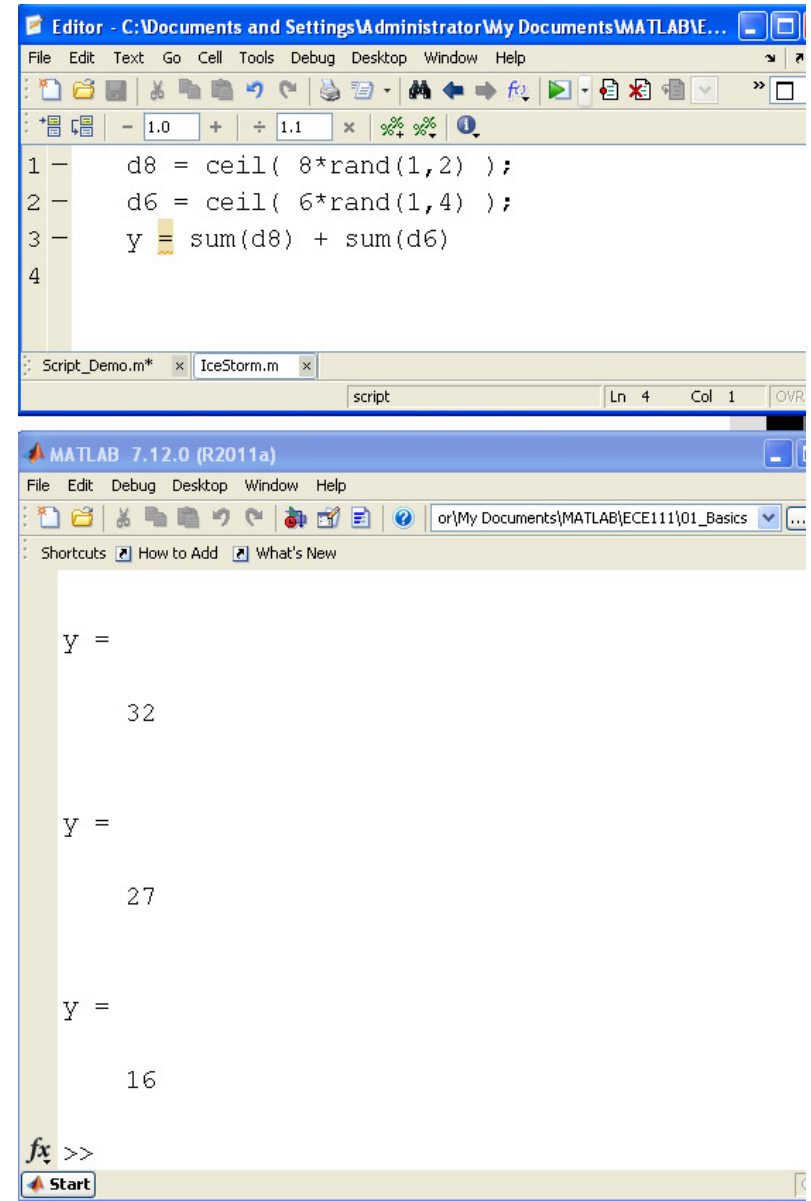
Example: Ice Storm

The Dungeons and Dragon's spell *Ice Storm* does 6-40 damage

- The sum of two 8-sided dice and four 6-sided dice
- $y = 2d8 + 4d6$

Determine...

- The probability of doing N damage
- The probability that $N > 30$



The image shows two windows from the MATLAB 7.12.0 (R2011a) environment. The top window is the MATLAB Editor, displaying a script named 'IceStorm.m' with the following code:

```
1 d8 = ceil( 8*rand(1,2) );  
2 d6 = ceil( 6*rand(1,4) );  
3 y = sum(d8) + sum(d6)  
4
```

The bottom window is the MATLAB Command Window, showing the results of the script's execution. It displays three separate outputs for the variable 'y':

```
y =  
32  
  
y =  
27  
  
y =  
16
```

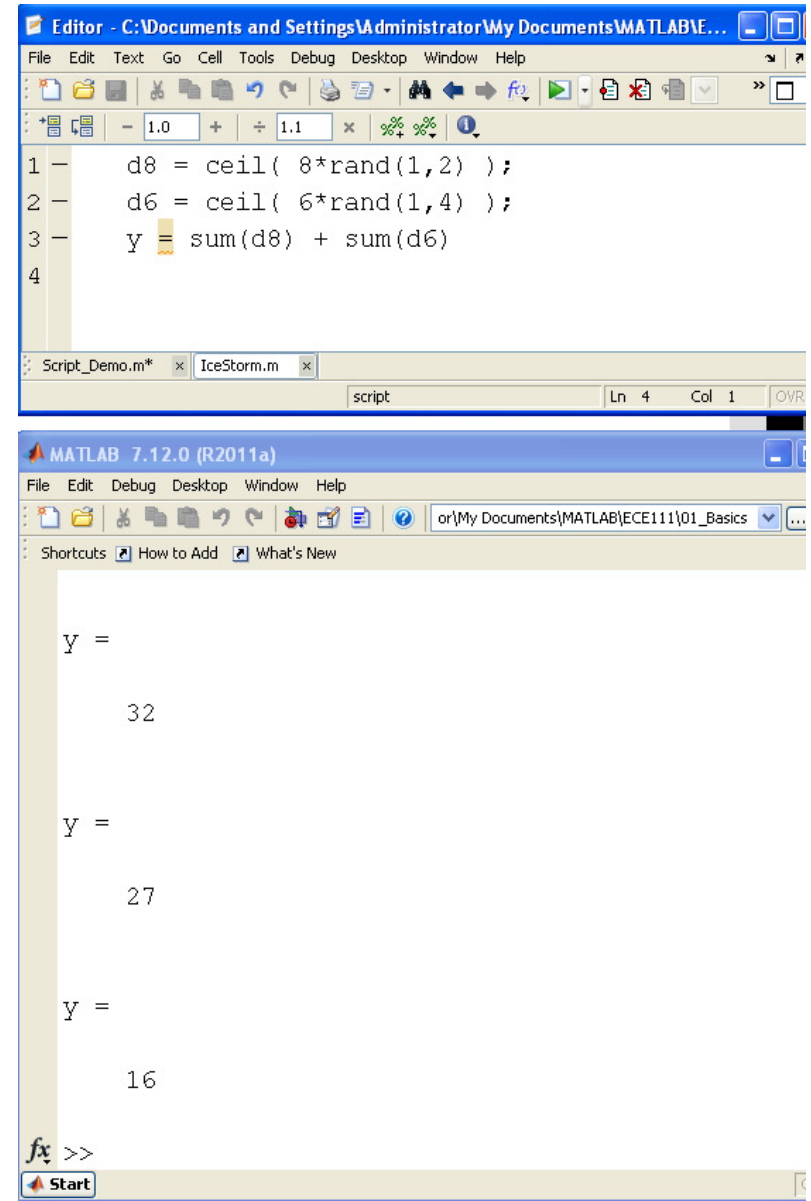
At the bottom of the Command Window, the prompt 'fx >>' is visible, indicating the end of the execution.

Solution: Step 1

- Create a file *IceStorm.m*
- Find $y = 2d8 + 4d6$

Note that every time you run this script, you get a different answer

- it's random



The image shows two windows from the MATLAB 7.12.0 (R2011a) environment. The top window is the MATLAB Editor, displaying a script named *IceStorm.m* with the following code:

```
1 d8 = ceil( 8*rand(1,2) );  
2 d6 = ceil( 6*rand(1,4) );  
3 y = sum(d8) + sum(d6)  
4
```

The bottom window is the MATLAB Command Window, showing the output of the script. It displays three separate runs of the script, each resulting in a different value for *y*:

```
y =  
32  
  
y =  
27  
  
y =  
16
```

The Command Window also shows the MATLAB prompt *fx >>* and a Windows taskbar at the bottom with the Start button.

Solution: Step 2

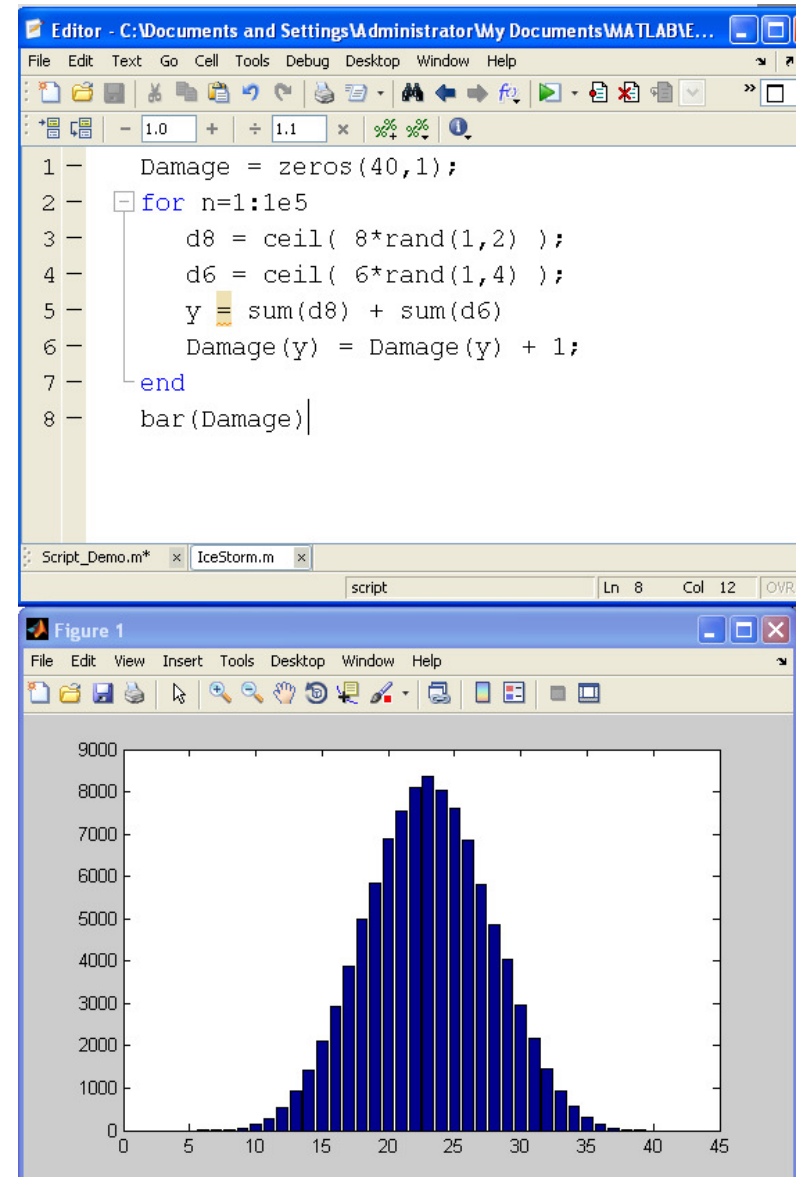
- Repeat 100,000 times
- Keep track of how many times you did y damage

The odds of doing 30 damage

```
>> Damage(30) / 100000  
ans = 0.0297
```

The odds of doing 30 or more damage

```
>> sum(Damage(30:40) / 100000)  
ans = 0.0866
```



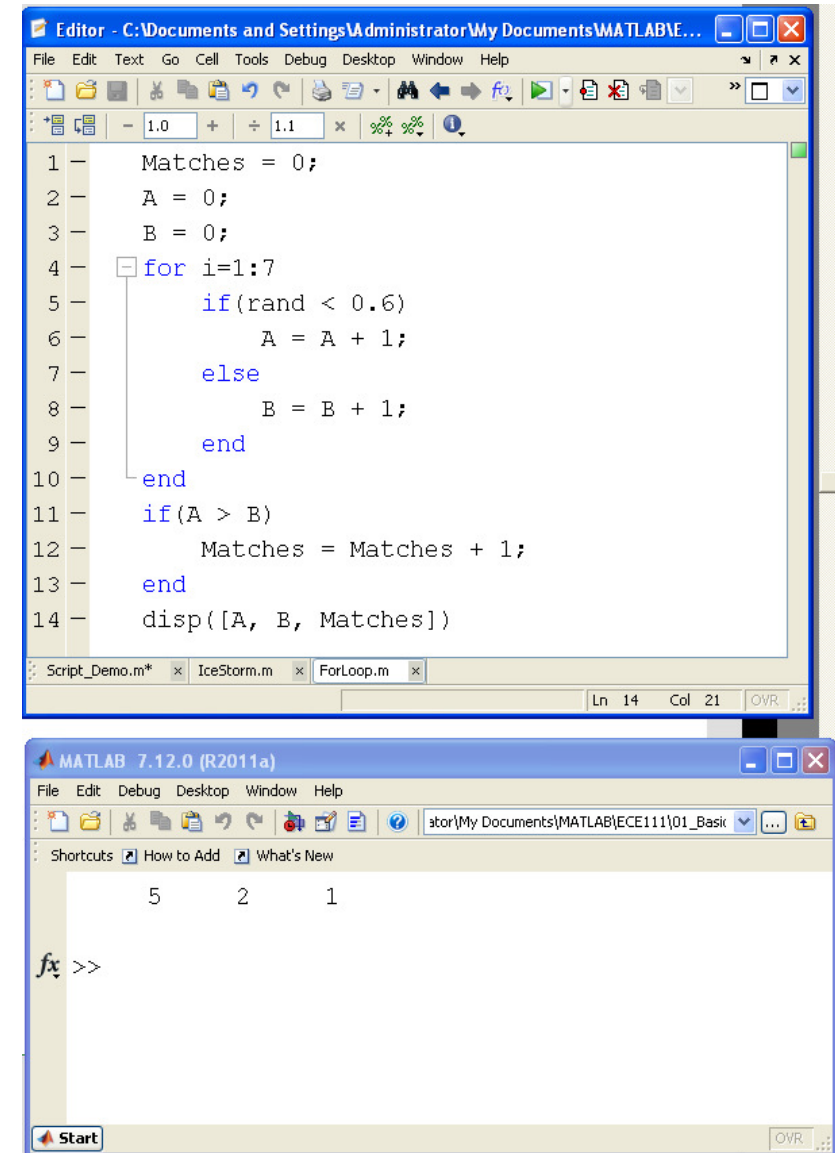
Monte-Carlo Example #2

A and B are playing a match

- A has a 60% chance of winning any given game.
- What is the probability that A will win the match?

Start by playing a single match

- A won the match 5-2
- Different result each time you run the script



The image displays two MATLAB windows. The top window is the MATLAB Editor, showing a script named 'Script_Demo.m' with the following code:

```
1 Matches = 0;
2 A = 0;
3 B = 0;
4 for i=1:7
5     if(rand < 0.6)
6         A = A + 1;
7     else
8         B = B + 1;
9     end
10 end
11 if(A > B)
12     Matches = Matches + 1;
13 end
14 disp([A, B, Matches])
```

The bottom window is the MATLAB Command Window, showing the output of the script:

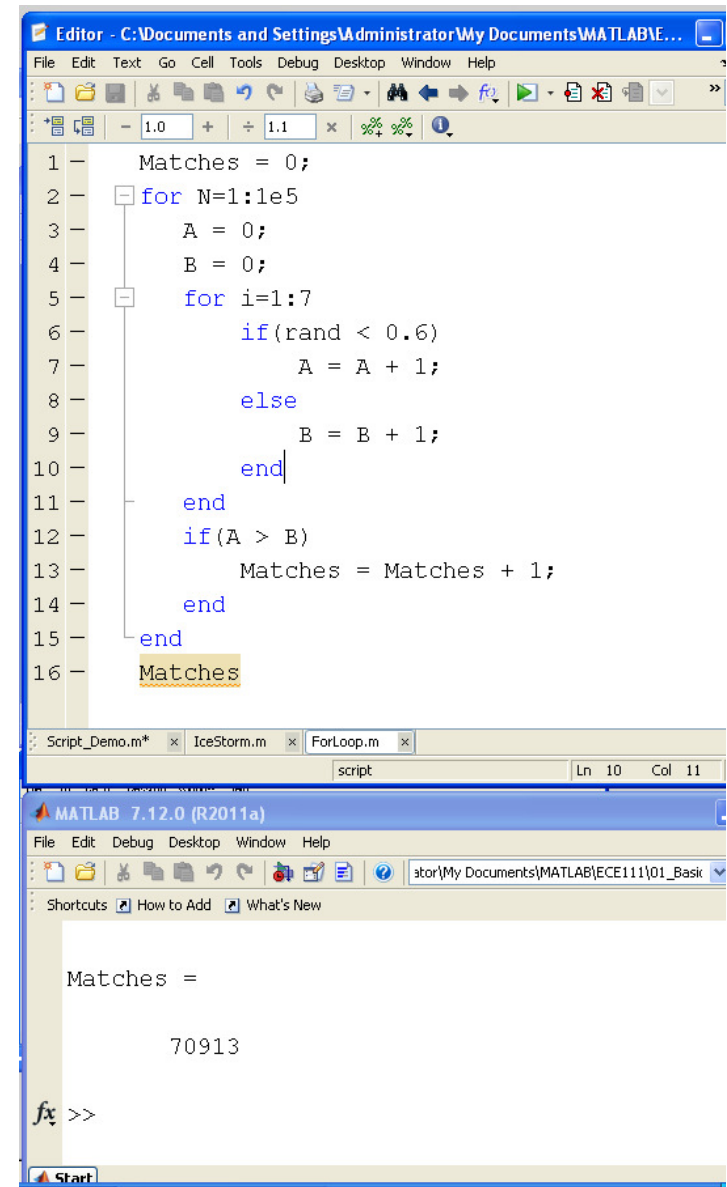
```
5     2     1

fx >>
```

The Command Window also shows the MATLAB logo and the prompt 'fx >>'.

Now repeat for 100,000 matches

- A wins 70,913 times in 100,000 matches
- A has roughly a 70.9% chance of winning any given match



The image shows a MATLAB 7.12.0 (R2011a) environment. The Editor window displays a script for simulating 100,000 matches. The script initializes 'Matches' to 0, then enters a loop for N=1 to 1e5. Inside this loop, it initializes A and B to 0, then enters a loop for i=1 to 7. In this inner loop, it uses 'rand' to generate a random number. If the number is less than 0.6, A is incremented by 1; otherwise, B is incremented by 1. After the inner loop, it checks if A is greater than B. If true, it increments 'Matches' by 1. Finally, it displays the value of 'Matches'.

```
1 Matches = 0;
2 for N=1:1e5
3     A = 0;
4     B = 0;
5     for i=1:7
6         if(rand < 0.6)
7             A = A + 1;
8         else
9             B = B + 1;
10        end
11    end
12    if(A > B)
13        Matches = Matches + 1;
14    end
15 end
16 Matches
```

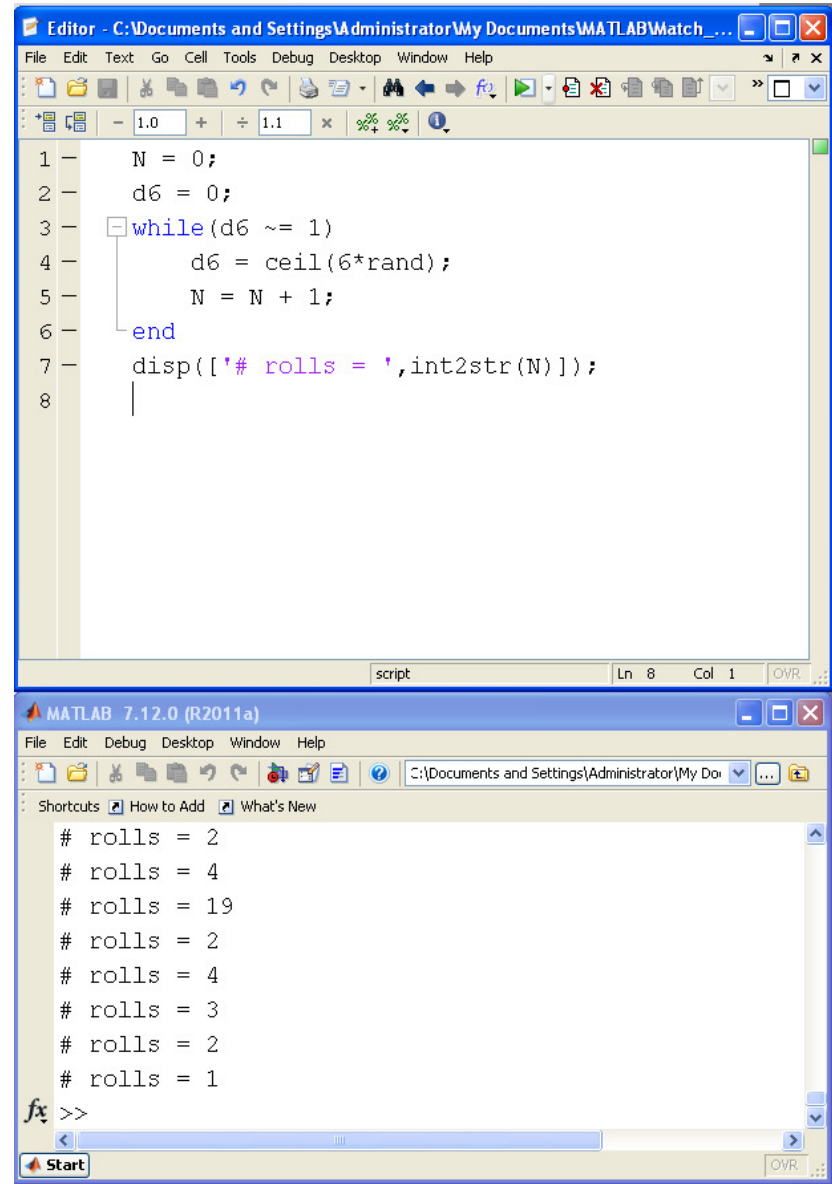
The Command Window shows the result of the script execution:

```
Matches =
70913
fx >>
```

While-Loop

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

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



The image displays two MATLAB windows. The top window is the MATLAB Editor, showing a script file named 'script'. The script contains a while loop that initializes a counter 'N' to 0 and a variable 'd6' to 0. The loop condition is 'd6 ~= 1'. Inside the loop, 'd6' is assigned the value of 'ceil(6*rand)', and 'N' is incremented by 1. The loop ends with 'end'. After the loop, the script displays the value of 'N' using 'disp(['# rolls = ',int2str(N)])'. The bottom window is the MATLAB Command Window, showing the output of the script: '# rolls = 2', '# rolls = 4', '# rolls = 19', '# rolls = 2', '# rolls = 4', '# rolls = 3', '# rolls = 2', and '# rolls = 1'. The Command Window also shows the MATLAB prompt 'fx >>' and a 'Start' button.

```
1 N = 0;
2 d6 = 0;
3 while(d6 ~= 1)
4     d6 = ceil(6*rand);
5     N = N + 1;
6 end
7 disp(['# rolls = ',int2str(N)]);
8
```

```
# rolls = 2
# rolls = 4
# rolls = 19
# rolls = 2
# rolls = 4
# rolls = 3
# rolls = 2
# rolls = 1
fx >>
```

While-Loop (cont'd)

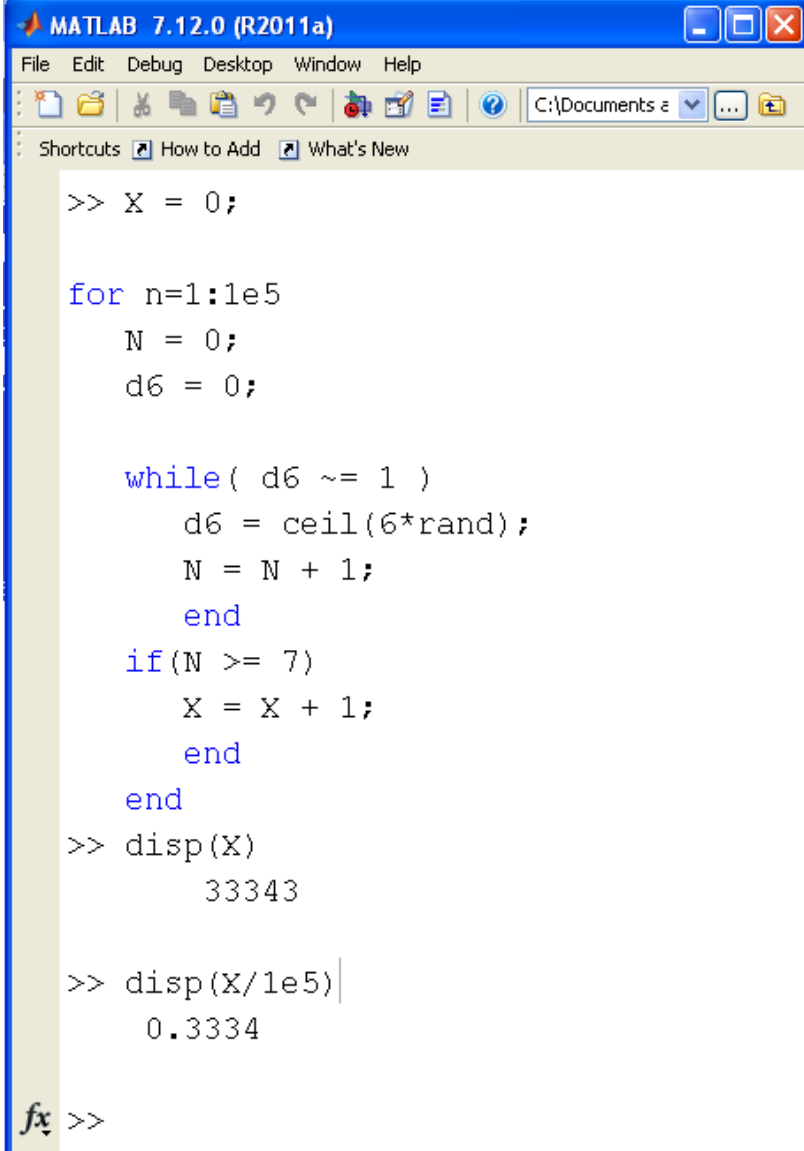
What is the chance that it will take 7 or more rolls to get a 1?

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



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Desktop Window Help
C:\Documents & Settings\user\My Documents\
Shortcuts How to Add What's New

>> X = 0;

for n=1:1e5
    N = 0;
    d6 = 0;

    while( d6 ~= 1 )
        d6 = ceil(6*rand);
        N = N + 1;
    end
    if(N >= 7)
        X = X + 1;
    end
end
>> disp(X)
    33343

>> disp(X/1e5)
    0.3334

fx >>
```

While-Loop (cont'd)

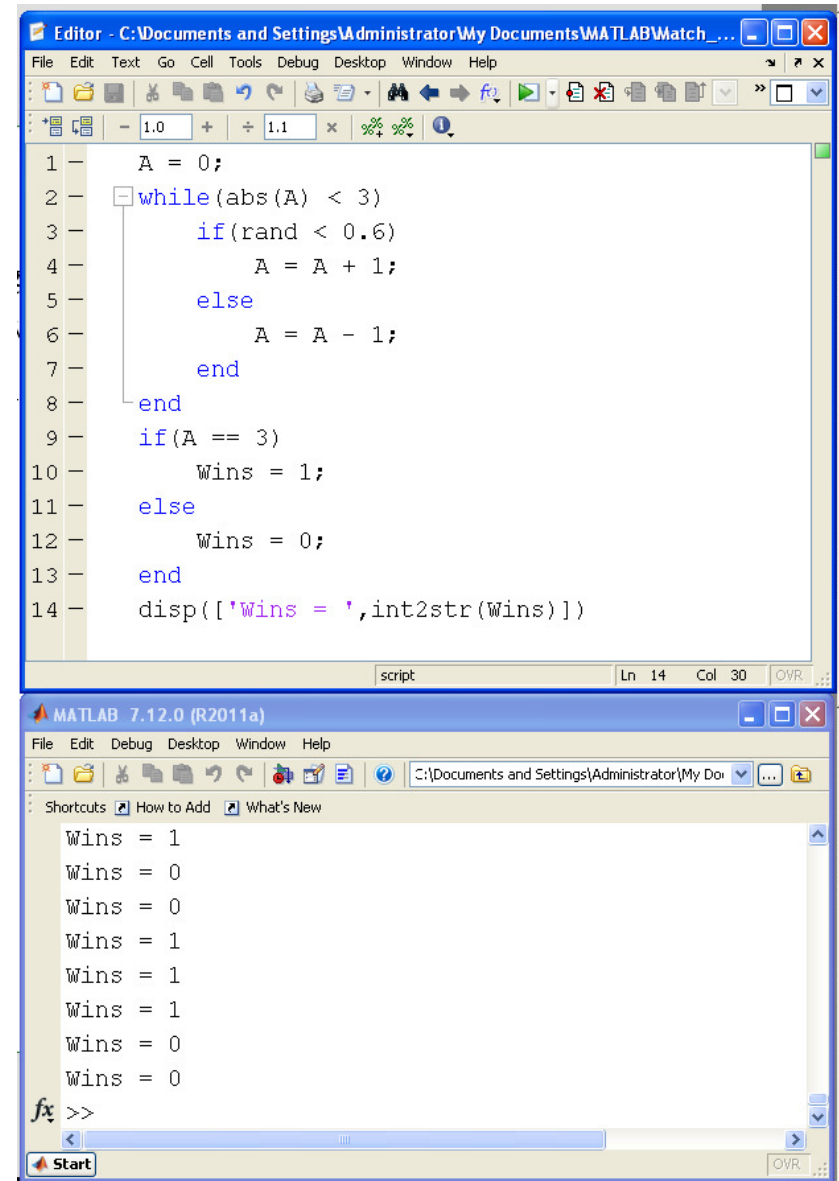
Player A and B are playing a match

- 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



The image shows two windows from the MATLAB 7.12.0 (R2011a) environment. The top window is the MATLAB Editor, displaying a script for simulating a match. The script initializes a variable A to 0 and enters a while loop that continues as long as the absolute value of A is less than 3. Inside the loop, a random number is generated; if it is less than 0.6, A is incremented by 1, otherwise it is decremented by 1. After the loop, the script checks if A equals 3. If true, it sets Wins to 1; otherwise, it sets Wins to 0. Finally, it displays the value of Wins. The bottom window is the MATLAB Command Window, showing the output of the script: a series of 'Wins = 1' and 'Wins = 0' statements, indicating the results of multiple simulations.

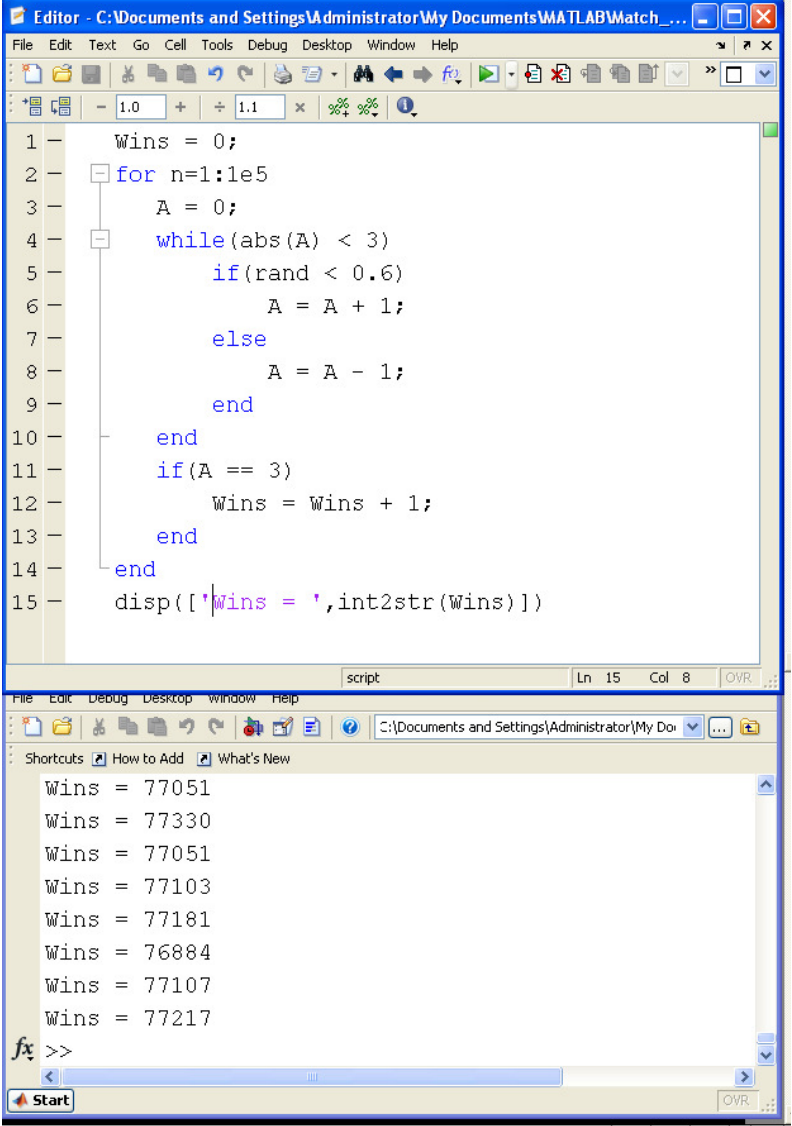
```
1 A = 0;
2 while(abs(A) < 3)
3     if(rand < 0.6)
4         A = A + 1;
5     else
6         A = A - 1;
7     end
8 end
9 if(A == 3)
10     Wins = 1;
11 else
12     Wins = 0;
13 end
14 disp(['Wins = ',int2str(Wins)])
```

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



The image shows a MATLAB Editor window and a Command Window. The Editor window displays a script for playing 100,000 matches. The Command Window shows the output of the script, which is the number of wins for player A after each match.

```
1 Wins = 0;
2 for n=1:1e5
3     A = 0;
4     while(abs(A) < 3)
5         if(rand < 0.6)
6             A = A + 1;
7         else
8             A = A - 1;
9         end
10    end
11    if(A == 3)
12        Wins = Wins + 1;
13    end
14 end
15 disp(['Wins = ',int2str(Wins)])
```

Wins = 77051
Wins = 77330
Wins = 77051
Wins = 77103
Wins = 77181
Wins = 76884
Wins = 77107
Wins = 77217

fx >>

While-Loop: Tennis

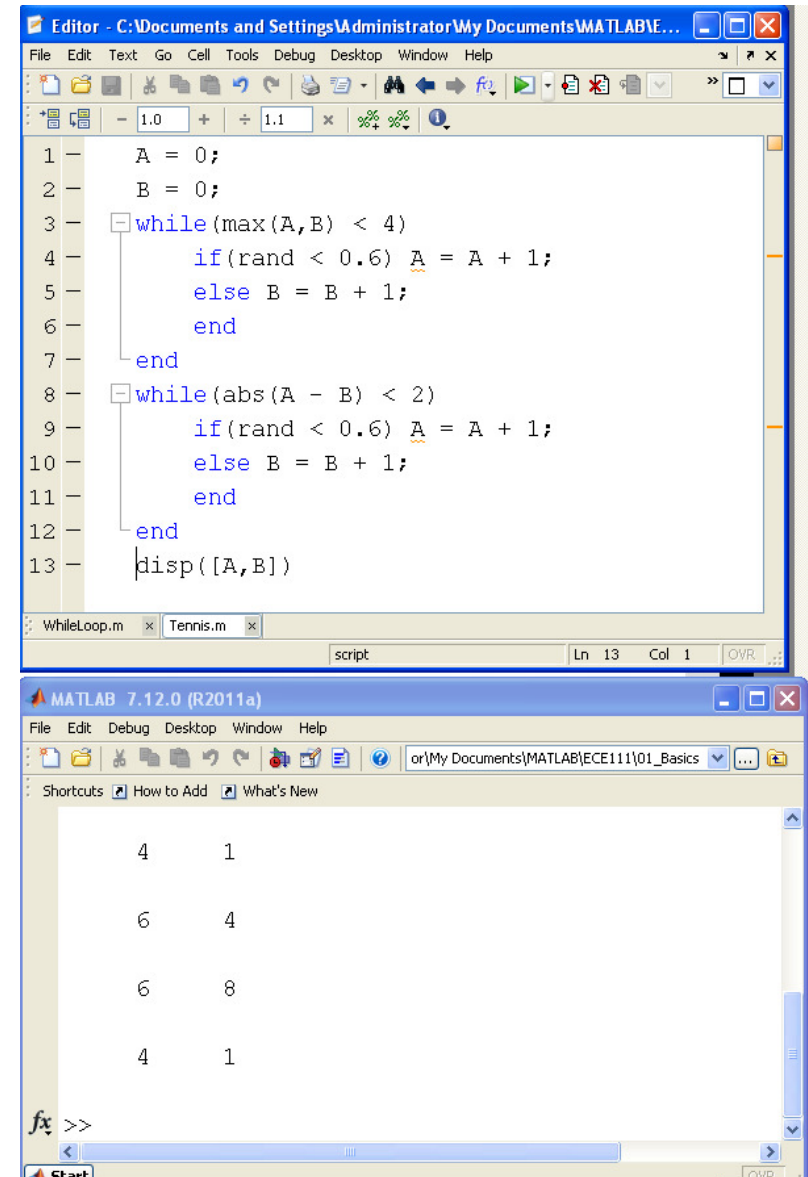
Assume A and B are playing a match

- A has a 60% chance of winning any given game
- If a player wins 4 games and is up by 2 games, the match is over.
- Otherwise, the match continues until a player is up two games.

Find the probability that A wins the match

Solution: Start with playing a single match

- A wins 4 - 1
- A wins 6 - 4
- A loses 8 - 6
- A wins 4 - 1



The image displays two MATLAB windows. The top window is the Editor, showing a script named 'Tennis.m' with the following code:

```
1 A = 0;
2 B = 0;
3 while(max(A,B) < 4)
4     if(rand < 0.6) A = A + 1;
5     else B = B + 1;
6     end
7 end
8 while(abs(A - B) < 2)
9     if(rand < 0.6) A = A + 1;
10    else B = B + 1;
11    end
12 end
13 disp([A,B])
```

The bottom window is the MATLAB Command Window, showing the output of the script as a 4x2 matrix:

4	1
6	4
6	8
4	1

The Command Window prompt is `fx >>`.

Now repeat for 100,000 matches

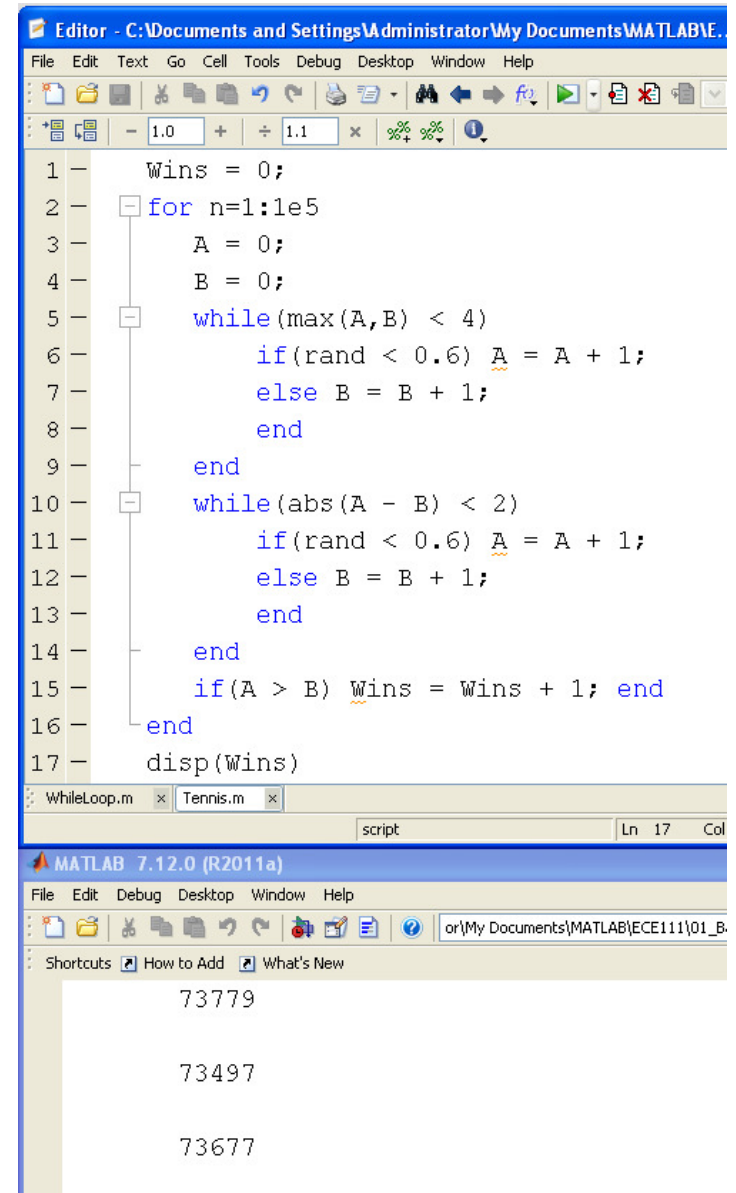
Result is A wins about 73,500 times

- A has about a 73.5% chance of winning the match

Results will vary each time you run this script

- it's random

To find the actual odds, you need to use a student-t test (week #15 of ECE 111)



The image shows a MATLAB 7.12.0 (R2011a) Editor window with a script titled 'Tennis.m'. The script simulates 100,000 matches between two players, A and B. Player A has a 60% chance of winning a point, and Player B has a 40% chance. The match ends when one player has a lead of 2 points. The script uses a for loop to repeat the simulation 100,000 times, and a while loop to simulate each match. The results are displayed in the Command Window.

```
1 Wins = 0;
2 for n=1:1e5
3     A = 0;
4     B = 0;
5     while(max(A,B) < 4)
6         if(rand < 0.6) A = A + 1;
7         else B = B + 1;
8     end
9 end
10 while(abs(A - B) < 2)
11     if(rand < 0.6) A = A + 1;
12     else B = B + 1;
13 end
14 end
15 if(A > B) Wins = Wins + 1; end
16 end
17 disp(Wins)
```

The Command Window shows the output of the script, which is the total number of wins for player A:

```
73779
73497
73677
```

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 π (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
 - linspace(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
-