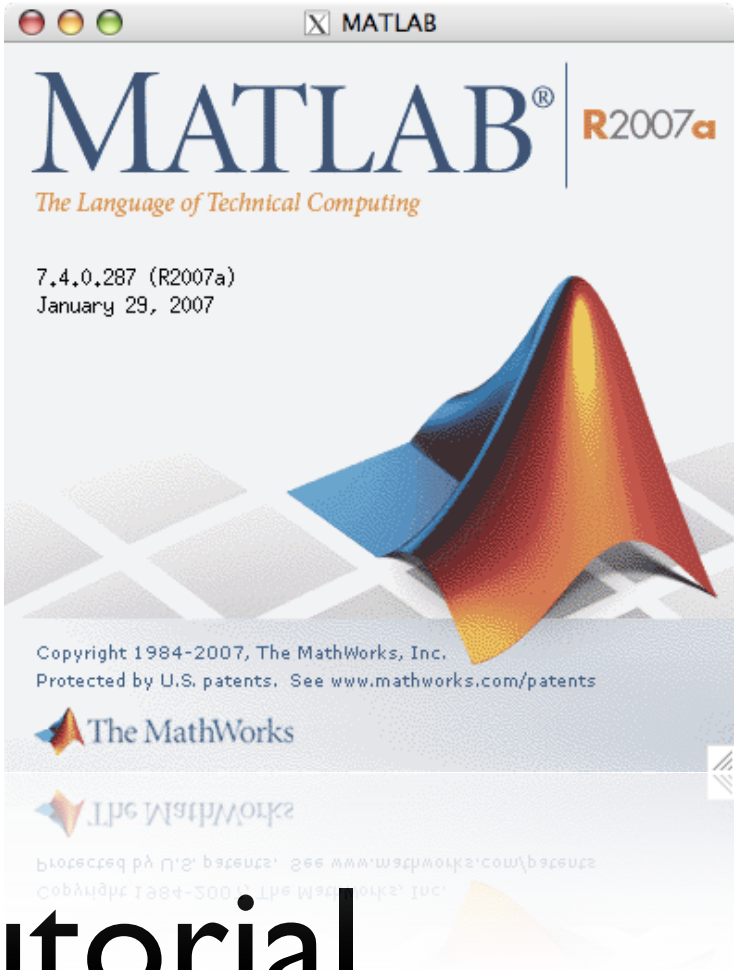


Matlab Tutorial

Joseph E. Gonzalez

What Is Matlab?

- MATrix LABoratory
  - Interactive Environment
  - Programming Language
- Invented in Late 1970s
  - Cleve Moler chairman CSD Univ New Mexico

Why we use it?

- Fast Development
- Debugging
- Mathematical Libraries
- Documentation
- Tradition
- Alternatives: Mathematica, R, Java? ML?...

Details

- Language
  - Like C and Fortran
  - Garbage Collected
- Interface
  - Interactive
  - Apple, Windows, Linux (Andrew)
  - Expensive ("Free" for you)

Matlab Language

Nap Time

- This is a comment
  - ((1+2)*3 - 2^2 - 1)/2
    - 2
- Use ; to suppress output (scripts and functions)
  - ((1+2)*3 - 2^2 - 1)/2;

- Assignment with equality
  - a = 5;
    - No Output
- Logical test like >, <, >=, <=, ~=
  - a == 6

- Short Circuited Logic
  - true || (slow_function)
    - 1 % Evaluates Quickly
  - true | (slow_function)
    - 1 % Evaluate slowly
- Matrix logic

- A simple array
  - [1 2 3 4 5]
    - 1 2 3 4 5
  - [1,2,3,4,5]
    - 1 2 3 4 5

- All the following are equivalent
  - [1 2 3; 4 5 6; 7 8 9]
  - [1,2,3; 4,5,6; 7,8,9]
  - [[1 2; 4 5; 7 8] [3; 6; 9]]
  - [[1 2 3; 4 5 6]; [7 8 9]]

- Creating all ones, zeros, or identity matrices
  - zeros(rows, cols)
  - ones(rows, cols)
  - eye(rows)
- Creating Random matrices

- Make a matrix
  - A = [1 2 3; 4 5 6; 7 8 9]
    - 1 2 3
      4 5 6
      7 8 9
- Access Individual Elements

Array and Matrix Indices Start at 1 not 0. (Fortran)

- Make a matrix
  - A = [1 2 3; 4 5 6; 7 8 9]
    - 1 2 3
      4 5 6
      7 8 9
- Access Individual Elements

- Make a matrix
  - A = [1 2 3; 4 5 6; 7 8 9]
    - 1 2 3
      4 5 6
      7 8 9

- Access Individual Elements
  - A(1, logical([1,0,1]))
    - 1 3
  - A(mod(A, 2) == 0)'
    - 4 2 8 6

- Make a matrix
  - A = [1 2 3; 4 5 6; 7 8 9]
    - 1 2 3
      4 5 6
      7 8 9
  - A + 2 * (A / 4)

- Make a matrix
  - A = [1 2 3; 4 5 6; 7 8 9]
    - 1 2 3
      4 5 6
      7 8 9
- Transpose
  - A’

- Matrix Multiplication
  - A*A % Equivalent to A^2
    - 30 36 42
      66 81 96
      102 126 150
- Element by Element Multiplcation
  - A .* A % equivalent to A.^2

- Matrix Multiplication
  - inv(A) % A^(-1)
    - 1.0e+16 *

0.3153	-0.6305	0.3153
-0.6305	1.2610	-0.6305
0.3153	-0.6305	0.3153

- Solving Systems

- Define some variables and store a function in f
  - c = 4;
  - f = @(x) x + c;
  - f(3)
    - 7

- Like arrays but can have different types
  - x = {‘hello’, 2, 3};
  - x{1}
    - ‘hello’
  - x{2}
    - 2

- Provide a convenient tool to organize variables
- Create Structs on the fly
  - point.x = 3;
  - point.y = 4;
  - point

Objects

- You can make objects but ...
  - you won't need them.
  - I don't know how to make them.
  - most people don't use them

- If Statements
  - c = rand();
  - if (c > .5) %% conditional

```
    disp('Greater than');
  elseif (c < .5)
    disp('Less Than');
  else
    disp('Equal to');
  end
```

- If Statements
  - count = 0;
  - for i = 1:length(data)

```
count = count + …
    (data(i,1) == 4 && data(i,3) == 2);
end
```

- Avoid using for loops

Scripts vs Functions

- Scripts
  - List of commands that operate on the current workspace
- Functions
  - List of commands that operate in a separate workspace
  - Takes in values from current workspace and returns values
  - Function name = filename
  - Can have additional (hidden) functions

my_script.m

```
disp(["x^2", …
   num2str(x^2)]);
y = x^2
```

my_fun.m

```
function [y, x] =
my_fun(x)
disp(["x^2", …
   num2str(x^2)]);
y=x^2
```

Functions must have same name as file.

my_script.m

```
y = x^2;
x = x + 3;
```

my_fun.m

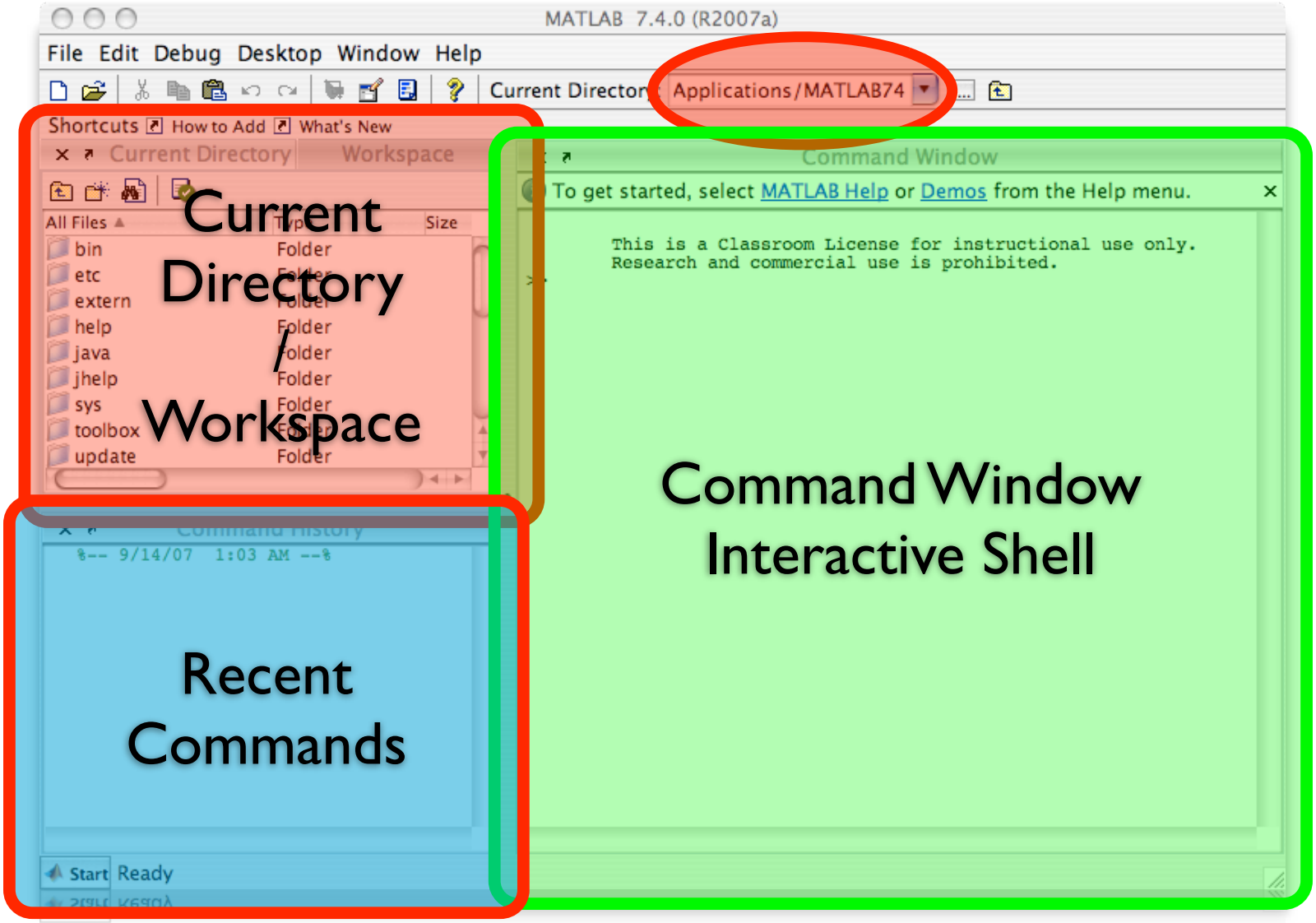
```
function [y, x] =
my_fun(x)
y=x^2;
x = x + 3;
```

- x=2; my_script;
- x
  - 5

```
>>  x=2; [y, xp] = my_fun(x);
>>  x
   ans:  2
>>  y
   ans:  4
>>  xp
   ans:  5
```

Things to Know

- Useful operators
  - >, <, >=, <=, ==, &, |, &&, ||, +, -, /, *, ^, …, ./, ‘, .*, .^, \
- Useful Functions
  - sum, mean, var, not, min, max, find, exists, clear, clc, pause, exp, sqrt, sin, cos, reshape, sort, sortrows, length, size, length, setdiff, ismember, isempty, intersect, plot, hist, title, xlabel, ylabel, legend, rand, randn, zeros, ones, eye, inv, diag, ind2sub, sub2ind, find,


MATLAB 7.4.0 (R2007a)
File Edit Debug Desktop Window Help
Current Director
Applications/MATLAB74
Shortcuts How to Add What's New
Current Directory
Workspace
All Files
Size
bin
etc
extern
help
java
jhelp
sys
toolbox
update
Folder
Command Window
To get started, select MATLAB Help or Demos from the Help menu.
This is a Classroom License for instructional use only.
Research and commercial use is prohibited.
%-- 9/14/07 1:03 AM --%
Start
Ready
Current Directory / Workspace
Command Window Interactive Shell
Recent Commands

Command Console

- Like a linux shell
  - Folder Based
  - Native Directories
  - ls, cd, pwd
- Use tab key to auto complete
- Use up arrow for last command

ls : List Directory Contents

```
>
README.txt    example3 tutorial.m
example1 my_function.m    tutorial1.m
example2 next.m       tutorial2.m
```

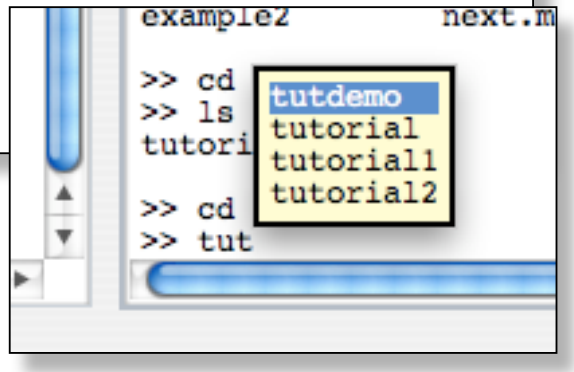
pwd :View Current directory

```
>
ans =
/Users/jegonzal/tutorial
```

cd : Change Directory

```
>> cd

>> pwd
ans =
/Users/jegonzal
```



- Get help on a function
  - help <function name>
- List names of variables in the environment
  - whos
- Clear the environment
  - clear

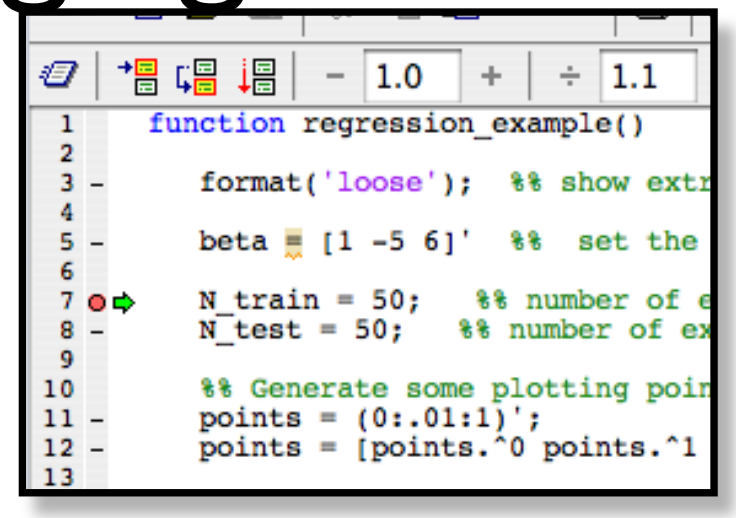
Folders

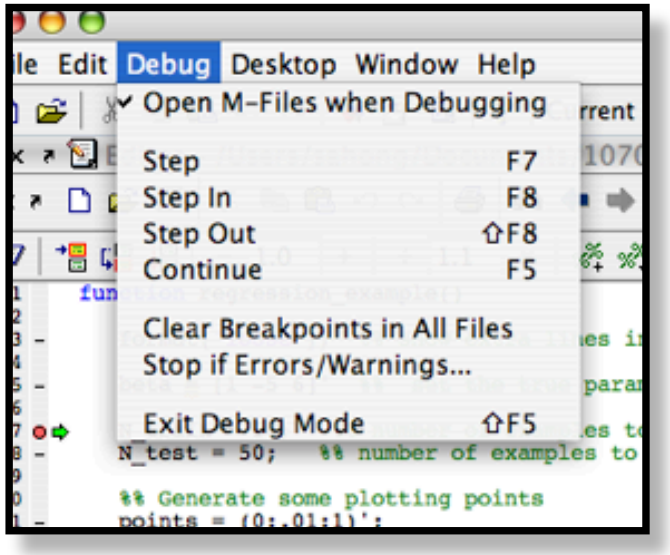
- Help organize your programs
- Can only call functions and scripts in:
  - The present working directory (pwd)
  - The Matlab path (path)
- Call functi yping name

```
>> my_script
>> y = my_function(x)
```

Debugging

- Insert break points
  - Click to the left of the line (Red Circle)
- Use interactive shell






```
K>>
K>> beta

beta =

     1
    -5
     6
```

Walk Through Interface