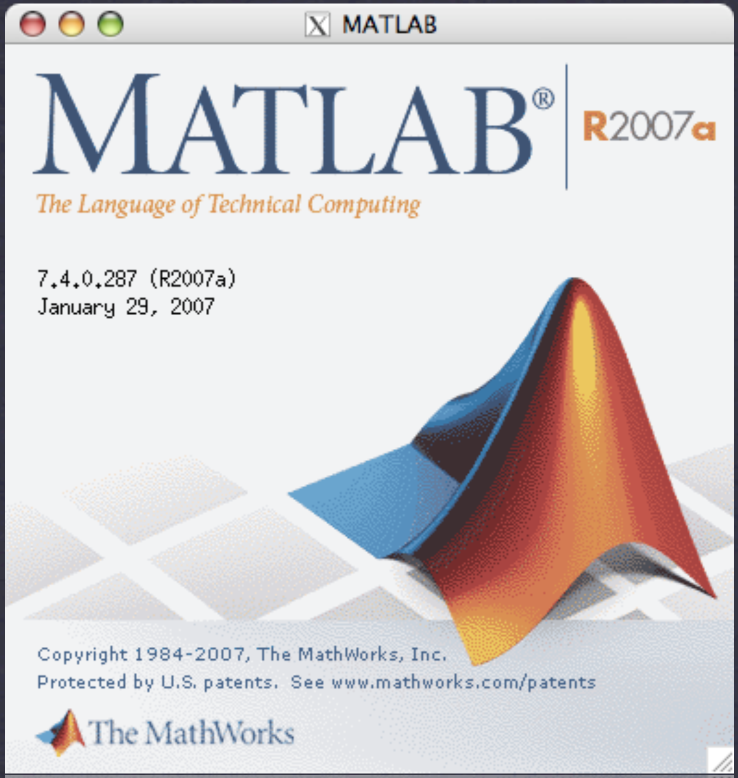


7.4.0.287 (R2007a)
January 29, 2007

Copyright 1984-2007, The MathWorks, Inc.
Protected by U.S. patents. See www.mathworks.com/patents

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
  - Fortran alternative to LINPACK
- Dynamically Typed, Garbage Collection

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)

Z Z Z Z

Matlab Language

Nap Time

Basics

```
% This is a comment
  >>  ((1+2)*3 - 2^2 - 1)/2
    ans:  2
% Use ; to suppress output (scripts and functions)
  >>  ((1+2)*3 - 2^2 - 1)/2;
      ❋ No output
% You need to use the ... operator to wrap lines
  >>  1 + 2 + 3 + 4 + 5 ...
       + 6 + 7 + 8 + 9
    ans:  45
```

Logic and Assignment

```
% Assignment with equality
  >>  a = 5;
```

- No Output

```
% Logical test like >, <, >=, <=, ~=
  >>  a == 6
    ans:  0  % 0 is false in Matlab (recall C)
  >>  a ~= 6
    ans:  1  % 1 is true in Matlab
```

- not(a == 6) also works

Logical Operators

```
% Short Circuited Logic
  >>  true || (slow_function)
    ans:  1  % Evaluates Quickly
  >>  true | (slow_function)
    ans:  1  % Evaluate slowly
% Matrix logic
  >>  matrix1 || matrix2
    ans:  Error
  >>  matrix1 | matrix2
```

- Pair wise logic

Making Arrays

```
% A simple array
  >>  [1 2 3 4 5]
     ans:  1  2  3  4  5
  >>  [1,2,3,4,5]
     ans:  1  2  3  4  5
  >>  1:5
     ans:  1  2  3  4  5
  >>  1:2:5
     ans:  1  3  5
  >>  5:-2:1
     ans:  5  3  1
```

Making Matrices

```
% 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]]
    ans:  1     2     3
          4     5     6
          7     8     9
```

More Making Matrices

```
% Creating all ones, zeros, or identity matrices
  >>  zeros( rows, cols )
  >>  ones( rows, cols )
  >>  eye( rows )
% Creating Random matrices
  >>  rand( rows, cols ) % Unif[0,1]
  >>  randn( rows, cols) % N(0, 1)
% Make 3x5 with N(1, 4) entries
  >>  1 + 2 * randn(3,5)
% Get the size
  >>  [rows, cols] = size( matrix );
```

Accessing Elements 1

```
% Make a matrix
   >>  A = [1 2 3; 4 5 6; 7 8 9]
      ans:  1   2   3
            4   5   6
            7   8   9
% Access Individual Elements
   >>  A(2,3)
      ans:  6
% Access 2nd column ( : means all elements)
   >>  A(:,2)
      ans:  2
            5
            8
```

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

Accessing Elements 2

```
% Make a matrix
  >>  A = [1 2 3; 4 5 6; 7 8 9]
     ans:  1   2   3
           4   5   6
           7   8   9
% Access Individual Elements
  >>  A([1, 3, 5])
     ans:  1   7   5
  >>  A( [1,3], 2:end )
     ans:  2   3
           8   9
```

Accessing Elements 3

```
% Make a matrix
  >>  A = [1 2 3; 4 5 6; 7 8 9]
    ans:  1    2    3
           4    5    6
           7    8    9
% Access Individual Elements
  >>  A(1, logical([1,0,1]))
    ans:  1    3
  >>  A( mod(A, 2) == 0)’
    ans:  4    2    8    6
```

```
>>  A(:)’
  ans:  1   4   7   2   5   8   3
    6   9
>>  A( mod(A, 2) == 0) = -1
  ans:  1   -1    3
       -1    5   -1
        7   -1    9
```

Matrix Math

```
% Make a matrix
  >>  A = [1 2 3; 4 5 6; 7 8 9]
     ans:  1   2   3
           4   5   6
           7   8   9
  >>  A + 2 * (A / 4)
     ans:  1.5000    3.0000    4.5000
           6.0000    7.5000    9.0000
          10.5000   12.0000   13.5000
  >>  A ./ A
     ans:  1    1    1
           1    1    1
           1    1    1
```

Matrix Math 2

```
% Make a matrix
  >>  A = [1 2 3; 4 5 6; 7 8 9]
    ans:  1   2   3
          4   5   6
          7   8   9
% Transpose
  >>  A'
    ans:  1   4   7
          2   5   8
          3   6   9
```

Matrix Math 3

```
% Matrix Multiplication
  >>  A*A % Equivalent to A^2
       ans:  30    36    42
              66    81    96
             102   126   150
% Element by Element Multiplcation
  >>  A .* A % equivalent to A.^2
       ans:  1     4     9
             16    25    36
             49    64    81
```

Matrix Inversion

```
% Matrix Multiplication
  >>  inv(A) % A^(-1)
     ans:  1.0e+16 *
         0.3153   -0.6305    0.3153
        -0.6305    1.2610   -0.6305
         0.3153   -0.6305    0.3153
% Solving Systems
  >>  (A + eye(3)) \ [1;2;3] % inv(A + eye(3)) * [1; 2; 3]
     ans:   -1.0000
            -0.0000
             1.0000
```

Anonymous Functions (Closure)

```
% Define some variables and store a function in f
  >>  c = 4;
  >>  f = @(x) x + c;
  >>  f(3)
     ans:  7
  >>  c = 5;
  >>  f(3)
     ans:  7
% This can be useful when you want to pass a function to a
  gradient library with the data already set.
```

Cells

```
% Like arrays but can have different types
  >>  x = {‘hello’, 2, 3};
  >>  x{1}
     ans:  ‘hello’
  >>  x{2}
     ans:  2
  >>  x{5} = @(x) x+1
     ans:  'hello'    [2]    [3]     []    @(x)x+1
  >>  x{5}(2)
     ans:  3
```

Structures

% Provide a convenient tool to organize variables

% Create Structs on the fly

```
>>  point.x = 3;

>>  point.y = 4;

>>  point

  ans:  point =
        x: 3
        y: 4
```

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

```
% If Statements
   >>   c = rand();
   >>   if (c > .5)    %% conditional
            disp(‘Greater than’);
         elseif (c < .5)
            disp(‘Less Than’);
         else
            disp(‘Equal to’);
         end
```

for statements

```
% If Statements
   >>  count = 0;
   >>  for i = 1:length(data)
           count = count + …
                (data(i,1) == 4 && data(i,3) == 2);
       end
% Avoid using for loops
   >>  count = sum( data(:,1) == 4 & data(:,3) == 2 )
% How would you compute the outer product of a row vector?
   >>  repmat(x, length(x), 1) .* repmat(x’, 1,length(x))
```

- Outer Product of row vector x

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

Files: Scripts and 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
% return;
end
```

Functions must have same name as file.

Pass by Value

my_script.m

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

```
>>  x=2;  my_script;
>>  x
  ans:  5
>>  y
  ans:  4
```

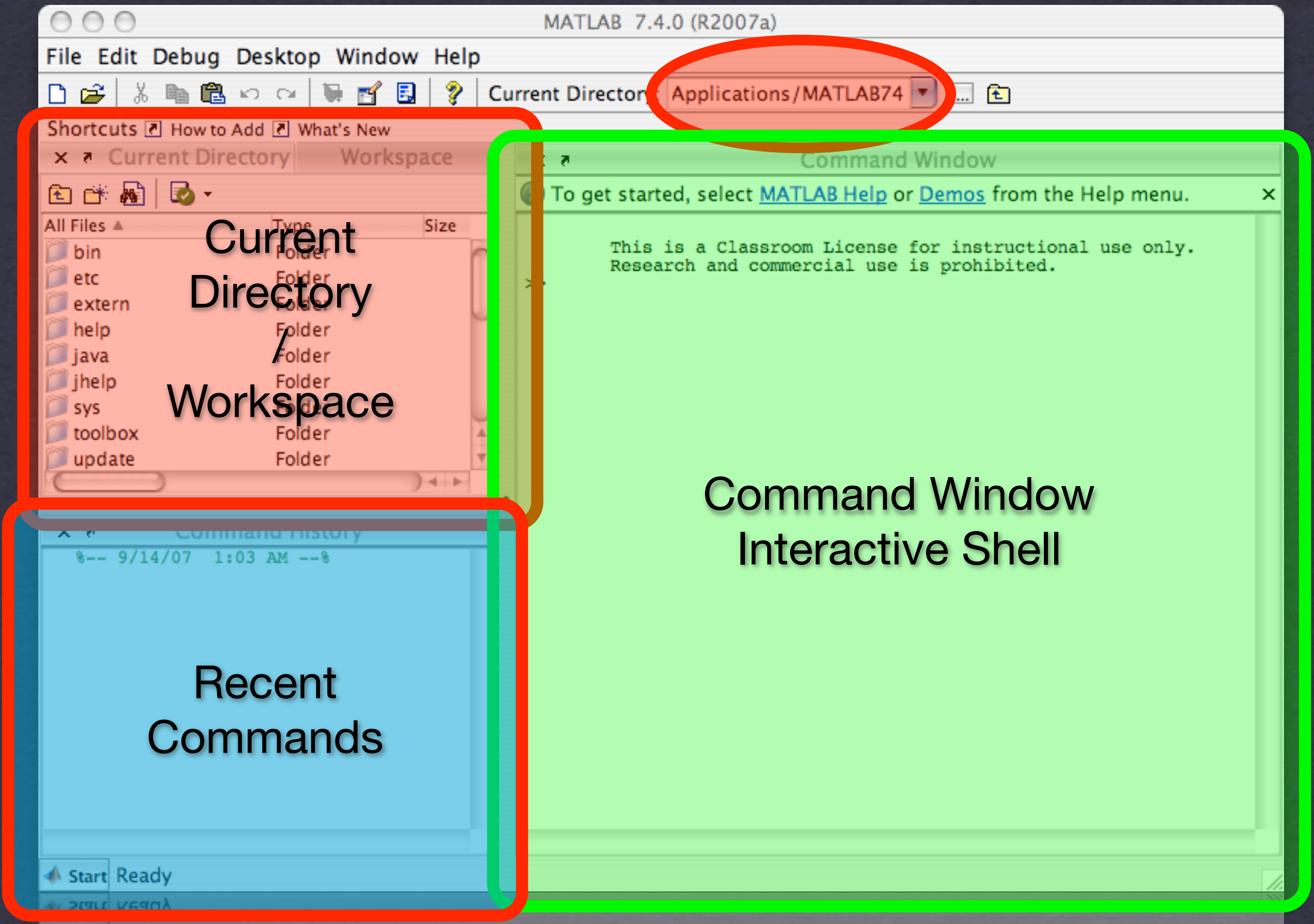
my_fun.m

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

```
>>  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, logical, repmat, num2str, disp, ...



To get started, select MATLAB Help or Demos from the Help menu.

```
%-- 9/14/07   1:03 AM --%
```

```
This is a Classroom License for instructional use only.
Research and commercial use is prohibited.
>>
```

Current Directory / Workspace

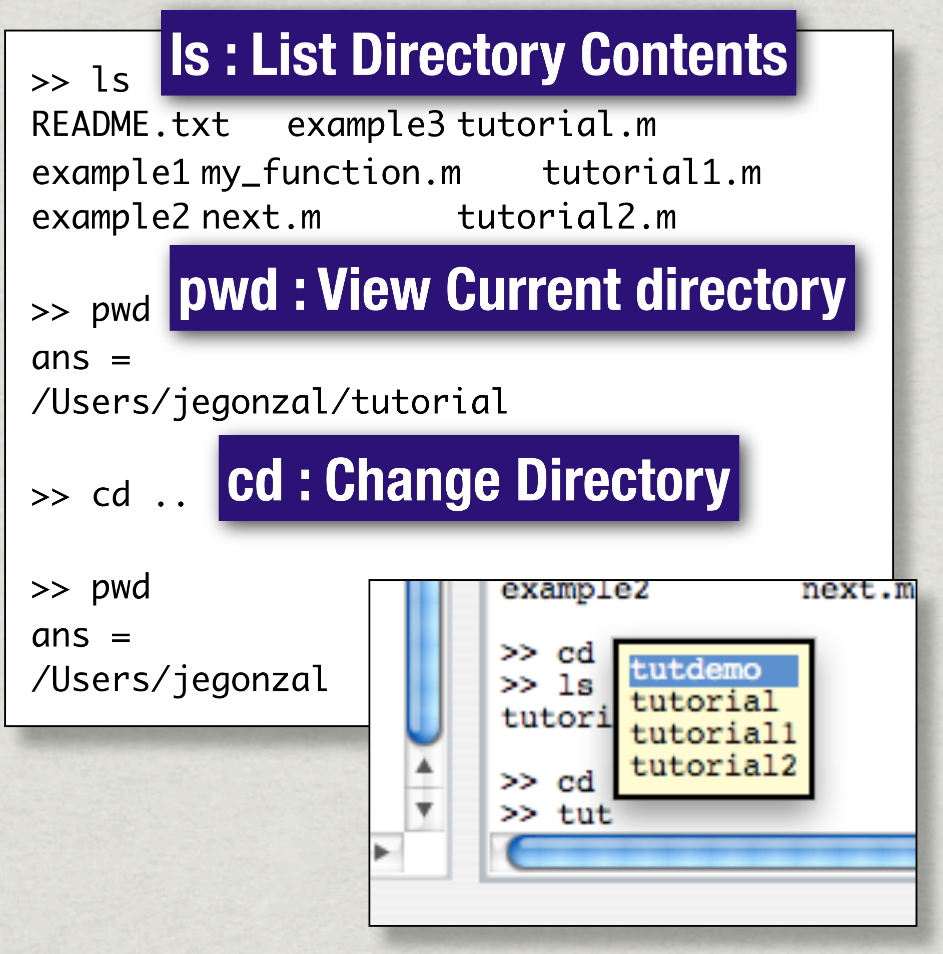
Command Window Interactive Shell

Recent Commands

THE INTERFACE

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

>> pwd
ans =
/Users/jegonzal/tutorial

>> cd ..

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

ls : List Directory Contents

pwd : View Current directory

cd : Change Directory

```
example2            next.m

>> cd
>> ls
tutori

>> cd
>> tut
```

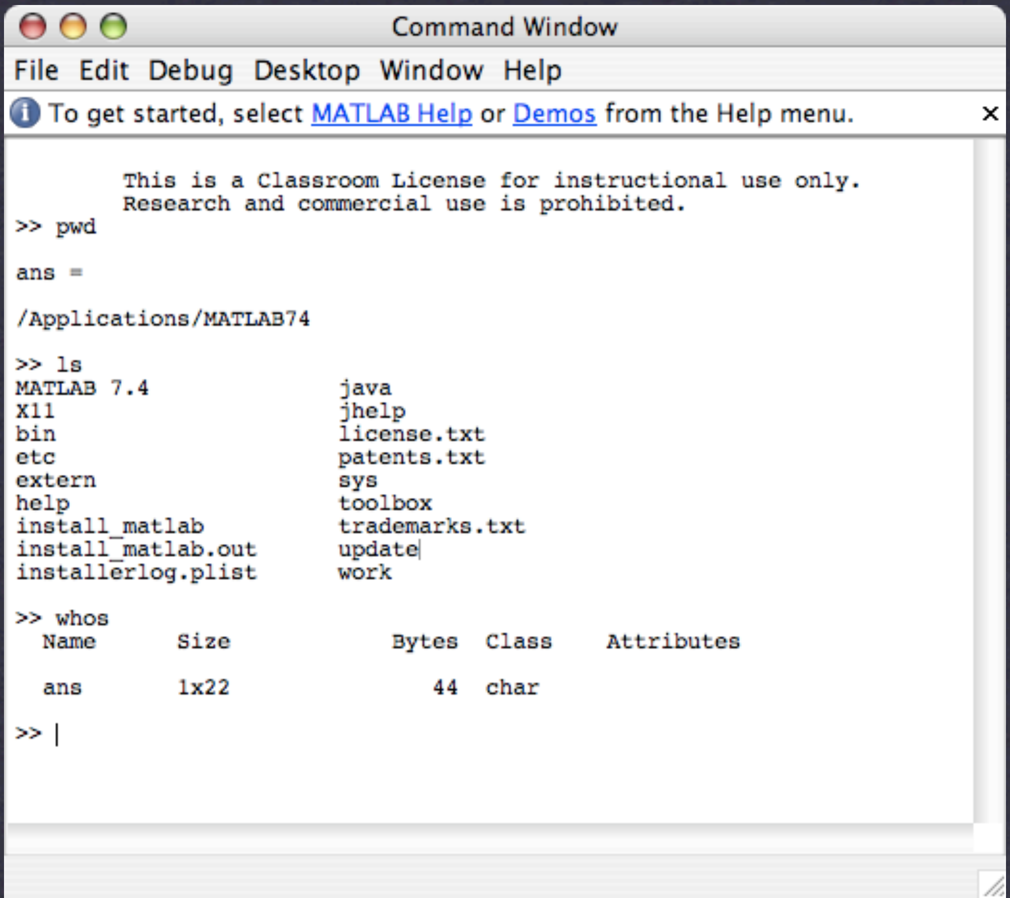
Other Commands

```
% Get help on a function
  >>  help <function name>
% List names of variables in the environment
  >>  whos
% Clear the environment
  >>  clear
% Edit functions and scripts
  >>  edit <filename>
% Open anything with the default “tool”
  >>  open <filename>
```

Folders

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

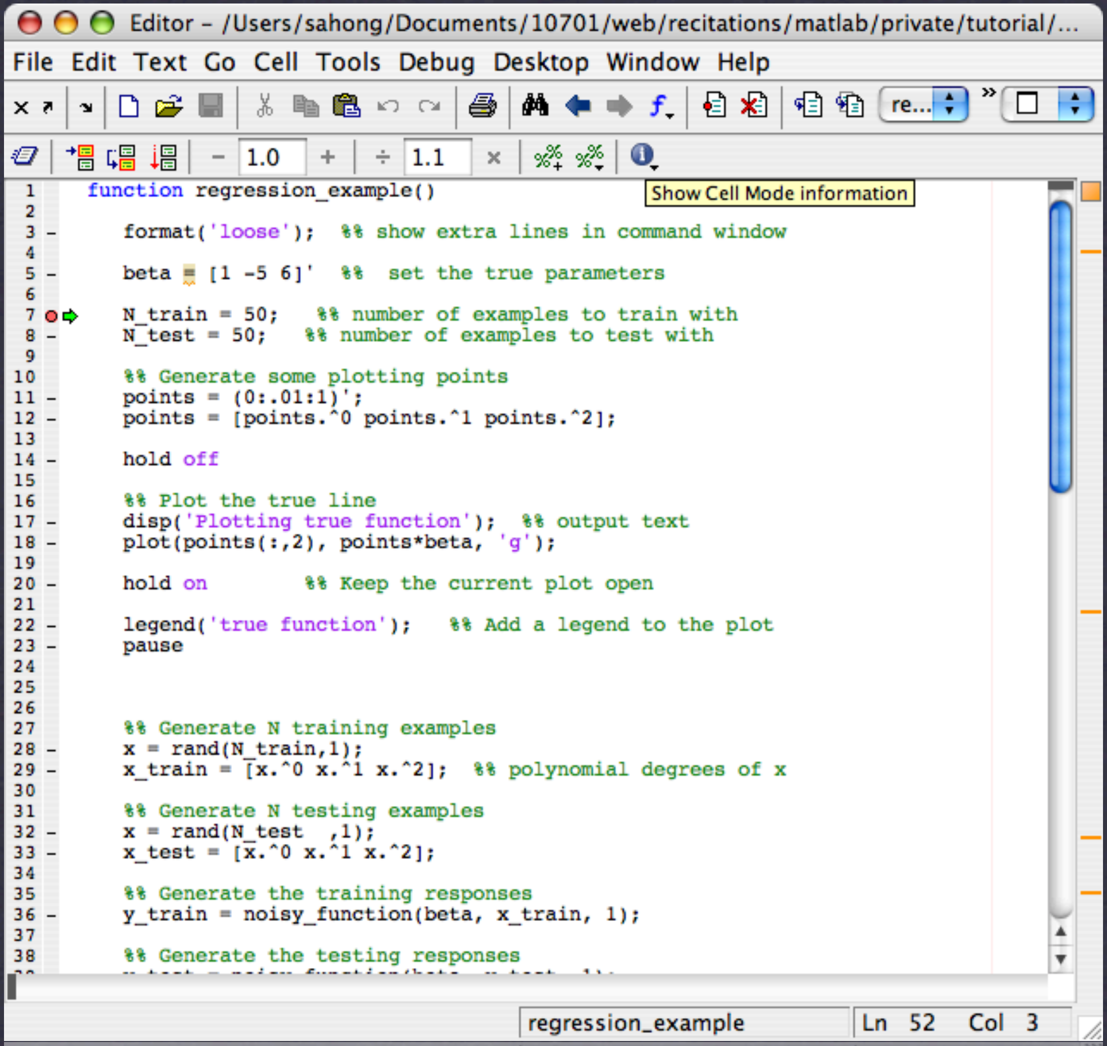
```
>> my_script
>> y = my_function(x)
```



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.
>> pwd

ans =

/Applications/MATLAB74

>> ls
MATLAB 7.4              java
X11                     jhelp
bin                     license.txt
etc                     patents.txt
extern                  sys
help                    toolbox
install_matlab          trademarks.txt
install_matlab.out      update
installerlog.plist      work

>> whos
  Name      Size            Bytes  Class    Attributes

  ans       1x22               44  char

>> 
```

GO PLAY WITH THE COMMAND WINDOW

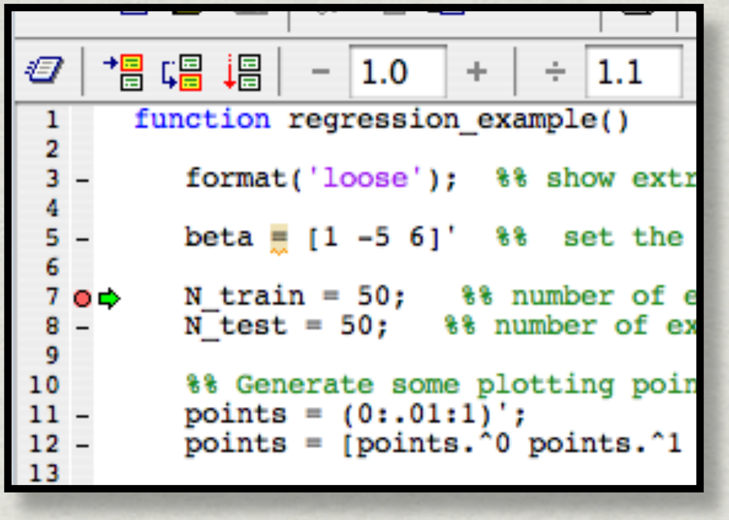

Editor – /Users/sahong/Documents/10701/web/recitations/matlab/private/tutorial/...
File Edit Text Go Cell Tools Debug Desktop Window Help
re...
1.0
1.1
Show Cell Mode information
regression_example
Ln 52 Col 3

```
function regression_example()

    format('loose');  %% show extra lines in command window

    beta = [1 -5 6]'  %%  set the true parameters

    N_train = 50;   %% number of examples to train with
    N_test = 50;   %% number of examples to test with

    %% Generate some plotting points
    points = (0:.01:1)';
    points = [points.^0 points.^1 points.^2];

    hold off

    %% Plot the true line
    disp('Plotting true function');  %% output text
    plot(points(:,2), points*beta, 'g');

    hold on         %% Keep the current plot open

    legend('true function');    %% Add a legend to the plot
    pause



    %% Generate N training examples
    x = rand(N_train,1);
    x_train = [x.^0 x.^1 x.^2];  %% polynomial degrees of x

    %% Generate N testing examples
    x = rand(N_test   ,1);
    x_test = [x.^0 x.^1 x.^2];

    %% Generate the training responses
    y_train = noisy_function(beta, x_train, 1);

    %% Generate the testing responses
    y_test = noisy_function(beta, x_test, 1);
```

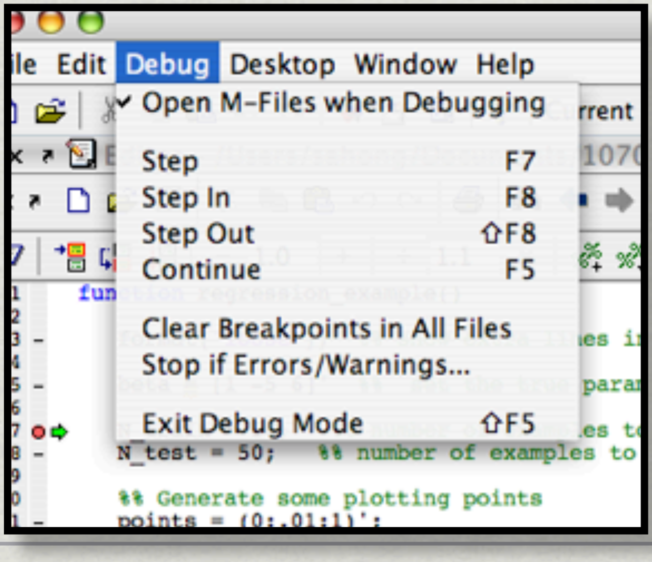
EDITOR

Debugging

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

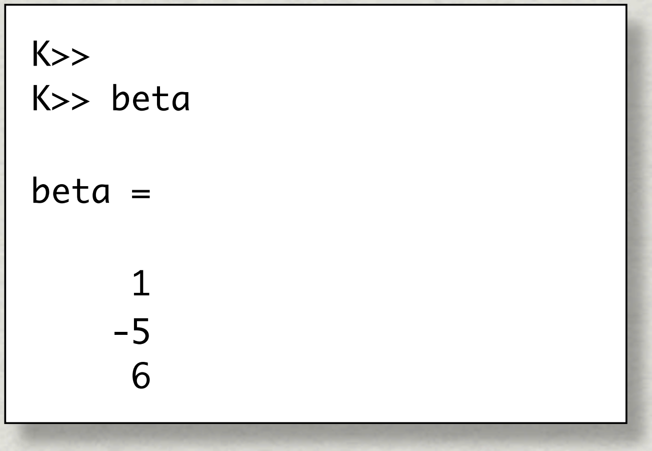


```
function regression_example()

    format('loose');  %% show extr

    beta = [1 -5 6]'  %%  set the

    N_train = 50;   %% number of e
    N_test = 50;   %% number of ex

    %% Generate some plotting poin
    points = (0:.01:1)';
    points = [points.^0 points.^1
```



le Edit Debug Desktop Window Help
✓ Open M-Files when Debugging
Step F7
Step In F8
Step Out ⇧F8
Continue F5
Clear Breakpoints in All Files
Stop if Errors/Warnings...
Exit Debug Mode ⇧F5

```
N_test = 50;    %% number of examples to

%% Generate some plotting points
points = (0:.01:1)';
```



```
K>>
K>> beta

beta =

     1
    -5
     6
```

Walk Through Interface