# MATLAB Course 2018-2019

Dr. Peter A. Bebbington

Brainpool AI

peter@brainpool.ai

bebbington @brainpoolai @brainpoolai

July 12, 2019

# ABOUT ME

- 2009: MPhys. Physics From The University of Manchester.
- 2009-2012: Freelance Software Engineer.
- 2012: MSc. Complex System Modelling From King's College London.
- 2013: MRes. Financial Computing From University College London.
- 2017: PhD. Physics From University College London.
- 2015-2017: Research Associate at a Multi-Strategy Hedge Fund.
- 2017-Present: CTO for an A.I. startup brainpool.ai

# OBJECTIVE

- Matlab Desktop
- Command Window
- Arrays
- Matrices
- Set Functions
- System of Linear Equations
- Plotting

# WHAT IS MATLAB?

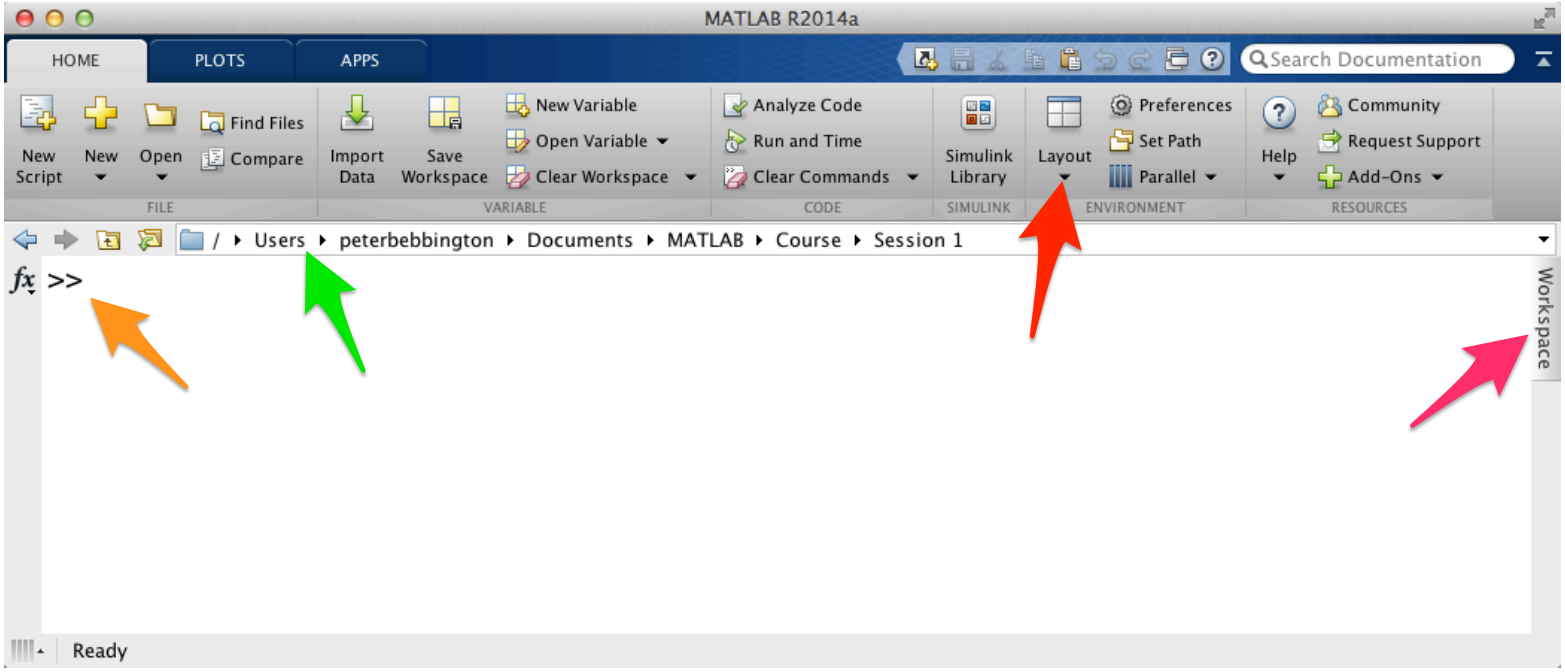
- MATLAB is an interpreted programming language. The statements are translated in to machine code one by one in MATLAB's interpreter. In comparison, a compiled programming language like C has a program translated as a whole into machine code by the compiler.
- There are three ways in which you can work in MATLAB. From the Command prompt or MATLAB, writing scripts and writing functions.
- A script file can take no arguments, it is just a series of statements that will be executed sequentially. A function file can take and return arguments.

# MATLAB DESKTOP

- The main MATLAB desktop has the following windows.
  - Command Window
  - Command History
  - Current Directory
  - Workspace
- MATLAB has extensive help files.
  - Text Based
  - Help Browser
  - Web Based: http://www.mathworks.com/

# MATLAB LAYOUT

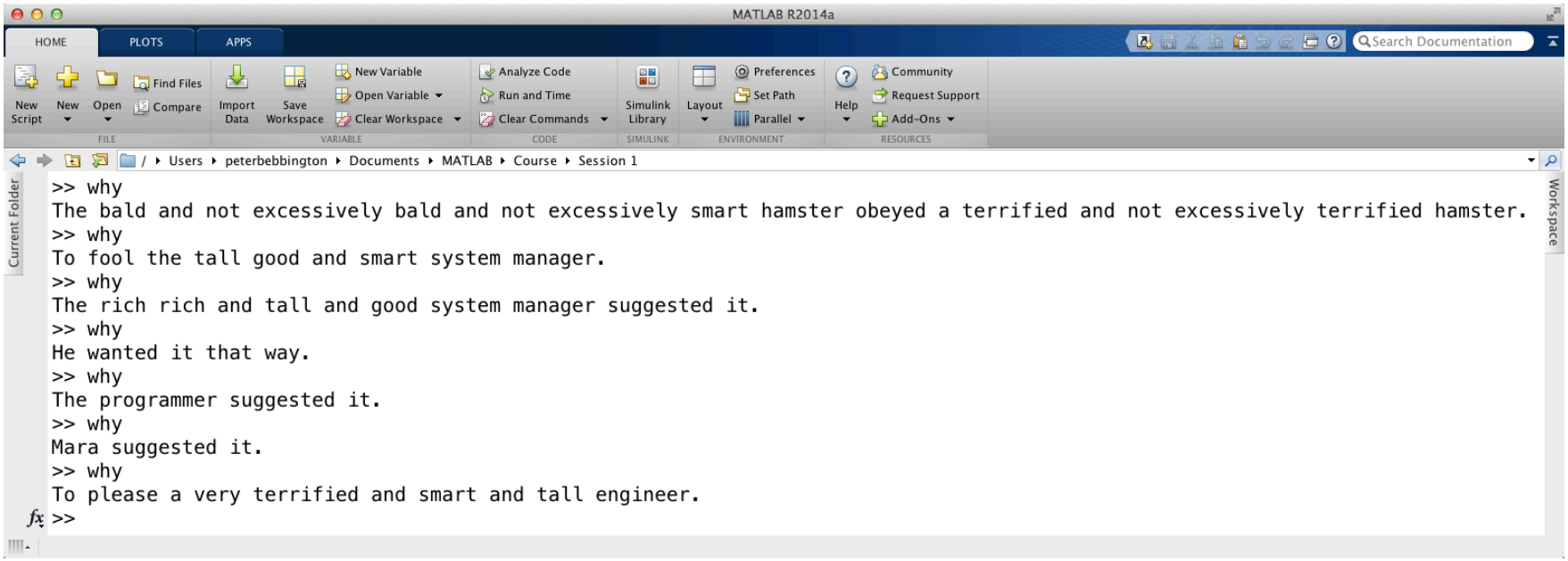
- Orange Arrow Command Line, Green Arrow Current Directory, Red Arrow Layout Editor and Pink Arrow Workspace.




- Makesure your current directory is something sensible for example folder on your desktop.

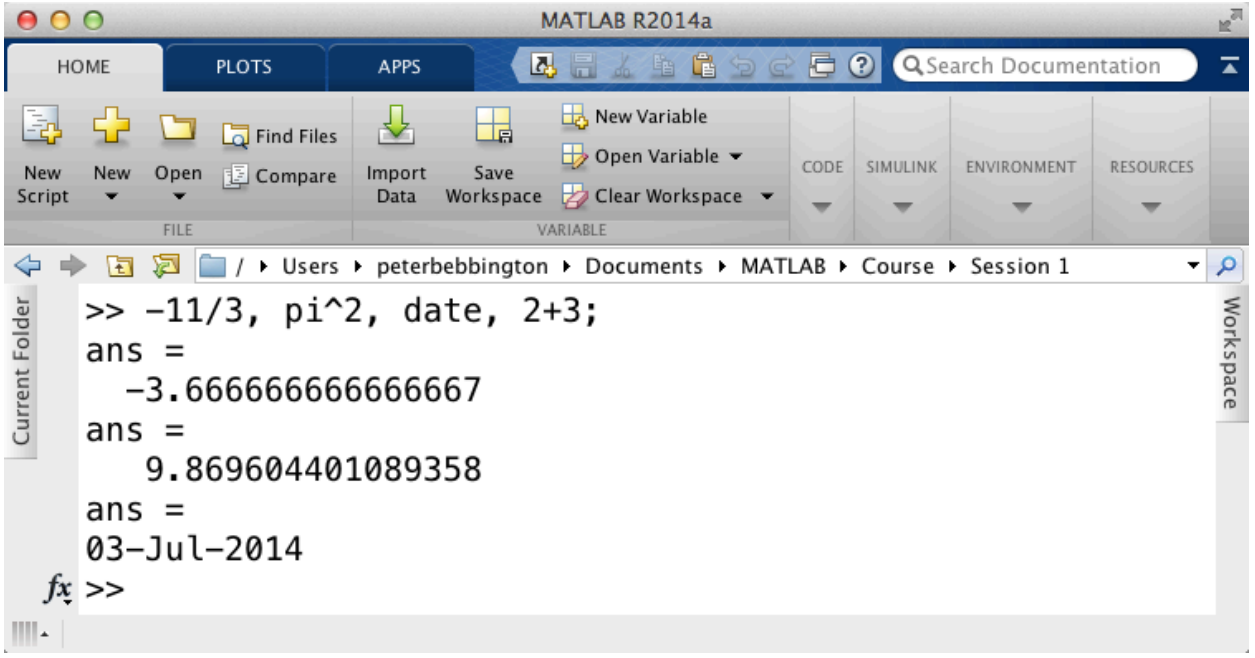
# COMMAND WINDOW

- Lets look at typing and returning a simple command in to the command prompt.



# NUMERICAL COMMANDS

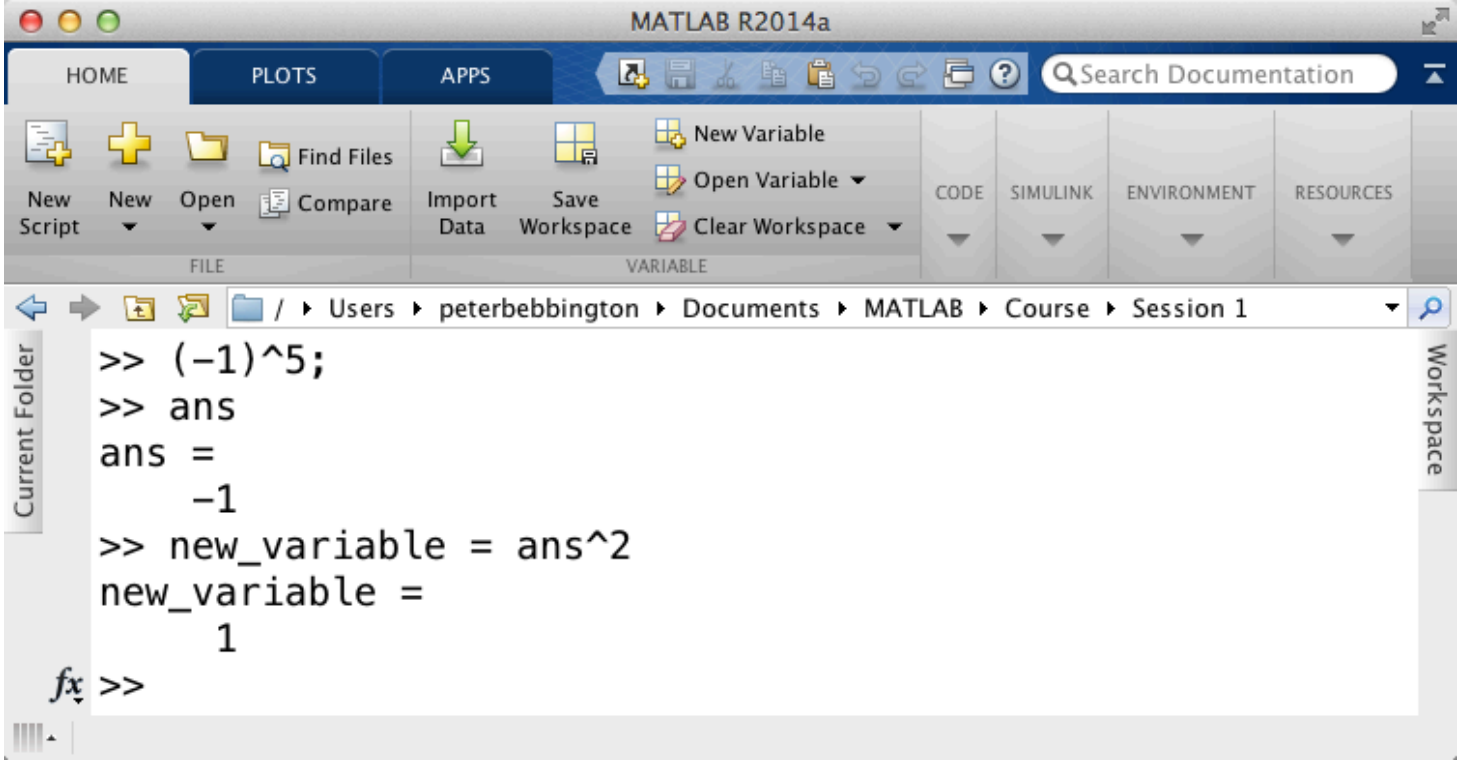
- Lets look at typing in some simple mathematical commands in the command prompt.




- For each command the output is creates the variable `ans` in the workspace.
- Notice that the semicolon stops the printing of the output to the command line and the comma allows the printing of the output to the command line.

# VARIABLES

- The last answer can be called by returning `ans`.
- One can assign a variable with = operator.

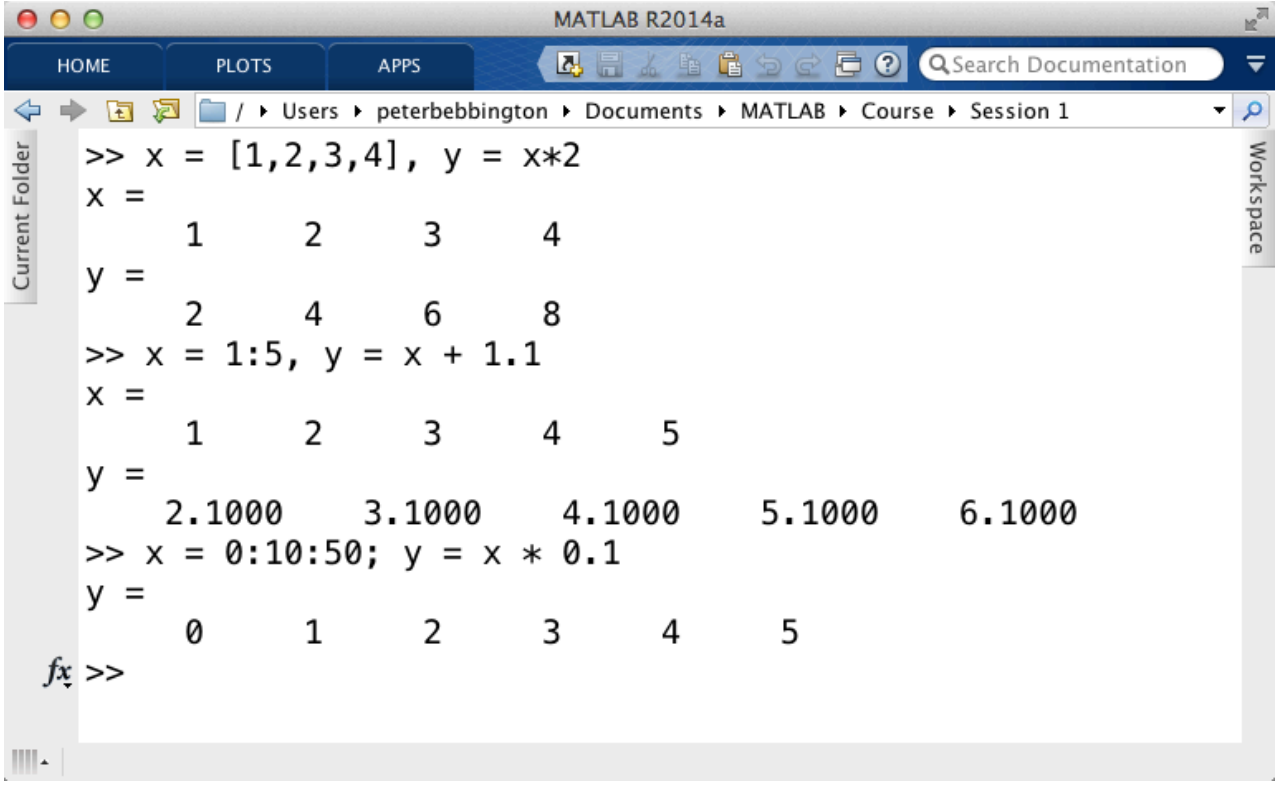


# COMMANDS AND SHORTCUTS

| Command | Meaning |
|---|---|
| `whos` | Gives the sizes and types of all loaded variables |
| `diary('filename')` | Save all commands in 'filename' |
| `diary on/off` | Set the mode of diary to on or off |
| `save 'workspace.mat'` | Save the current workspace in '`workspace.mat`' |
| `load 'workspace.mat'` | Load '`workspace.mat`' into the current workspace |
| `close all` | Closes all figure windows |
| `clear all` | Clears from memory all loaded variables |
| `clc` | Clear command windows |
| `ctrl+c` | Stops execution of a programme |
| `ctrl+]` or `[` / ⌘+] or [ | Indent the selected text forward or backward |
| `ctrl+R` or `T` / ⌘+/ or T | Comment or uncomment selected text |
| ↑ ↓ | Helps to search through old commands |
| `quit` or `exit` | Exit Matlab |

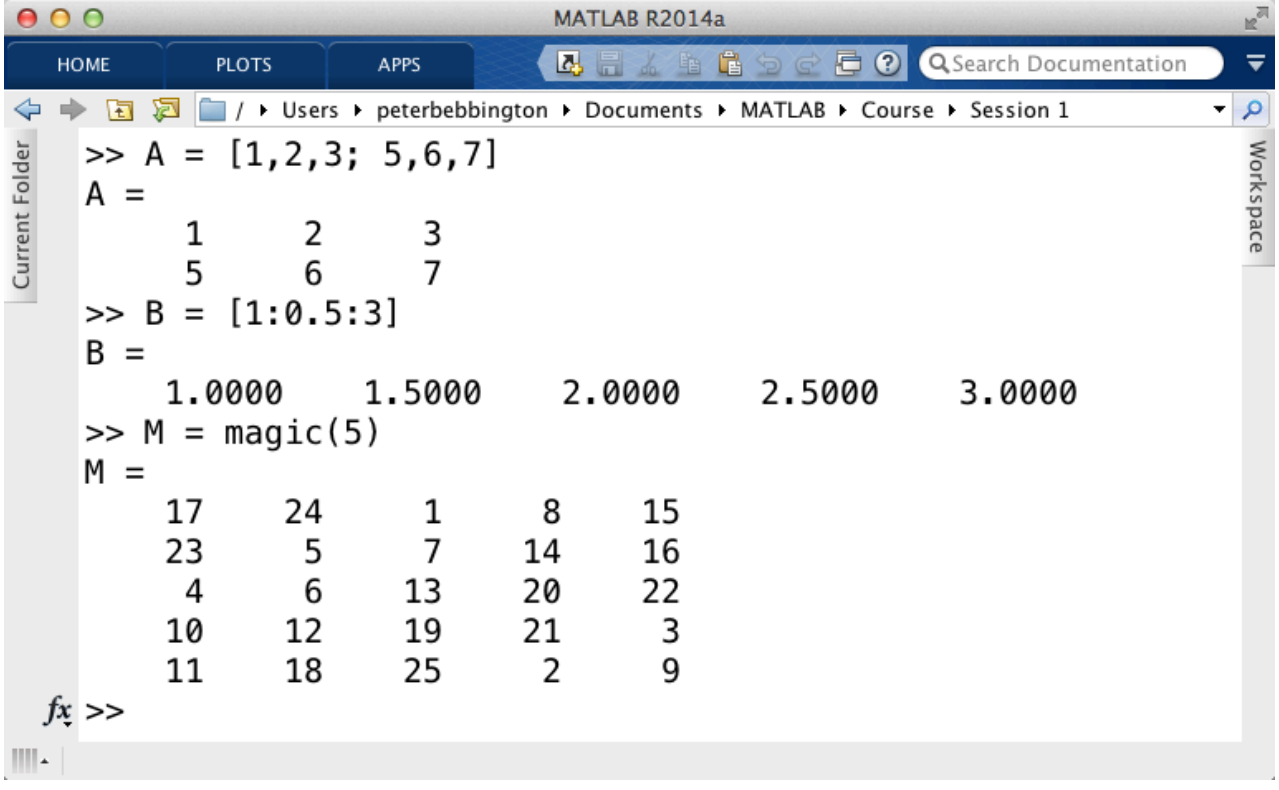
# ARRAYS

- Arrays are equivalent to vectors.
- Lets create an array of homogeneous variables (vectors) and operate on collectively on elements.



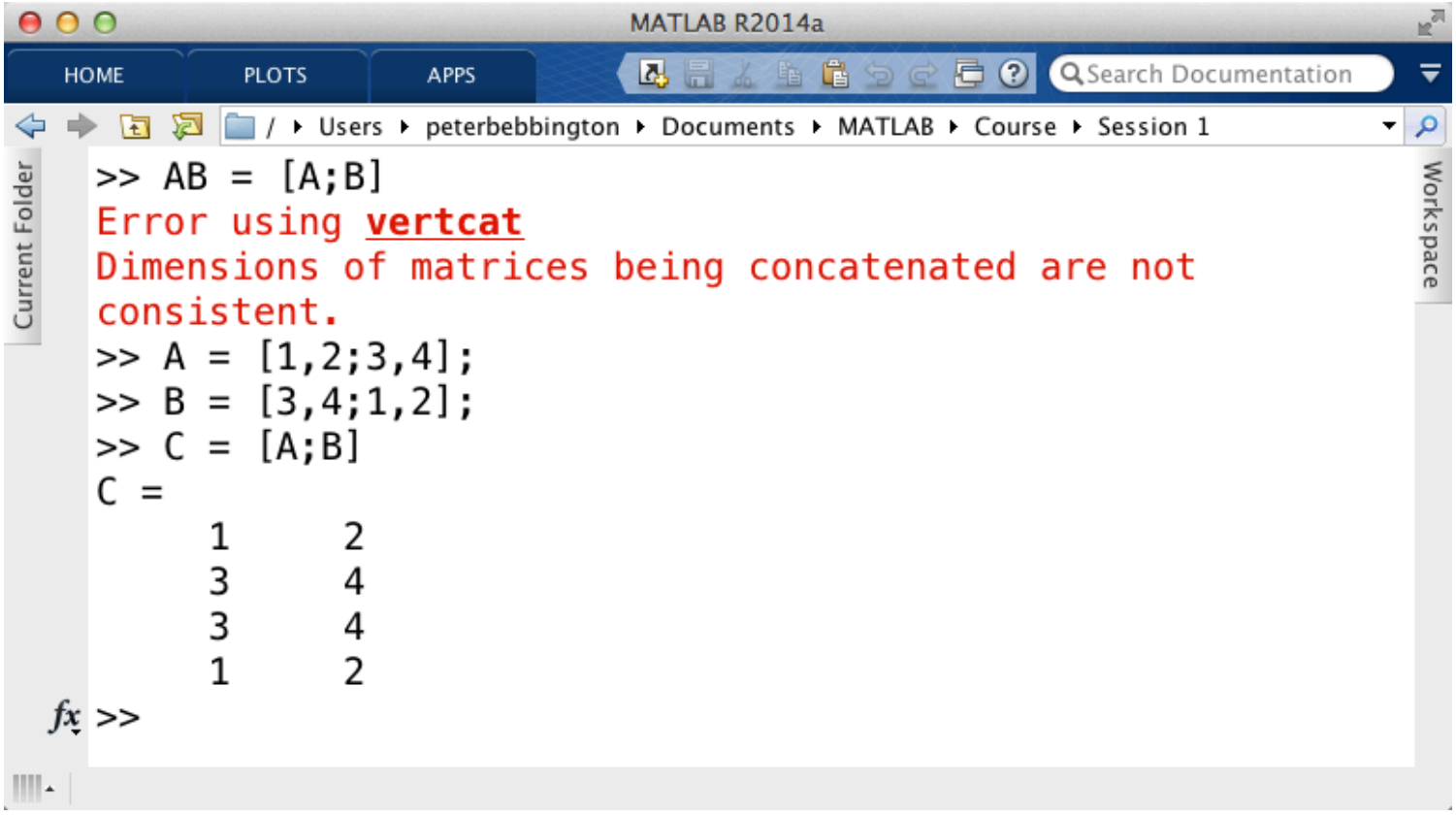
# BUILDING MATRICES

- Matrix Constructor:
  - Column Separator ,
  - Row Separator ;
  - Uniformly Spaced :



# CONCATENATING

- Matrices can be combined if and only if they have the correct dimensions.
- To do this one uses the [] operator.



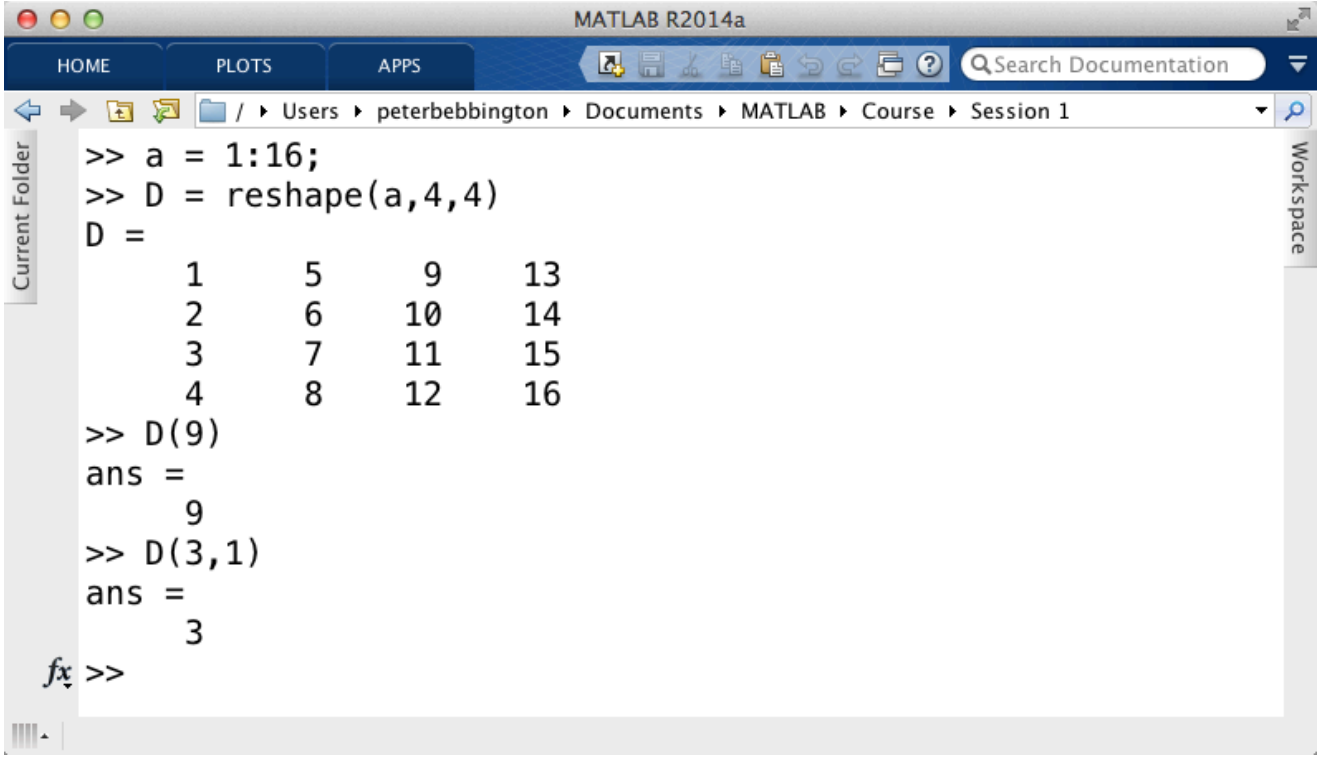
# ADDRESSING MATRIX ELEMENTS

- Round Brackets are used to address Matrix elements: D(r,c).
- It is also possible to address matrix elements via vectors: D([r, c]).

| | | | |
|---|---|---|---|
| 1,1 | 1,2 | 1,... | 1,$c_n$ |
| 2,1 | 2,2 | 2,... | 2,$c_n$ |
| ...,1 | ...,2 | ...,... | ...,$c_n$ |
| $r_m$,1 | $r_m$,2 | $r_m$,... | $r_m$,$c_n$ |

# MATRIX ELEMENTS EXAMPLE

- Notice we have used the `reshape` function which can take an array as an augment and change the shape to another array with consistent dimensions.



# MATRIX OPERATIONS

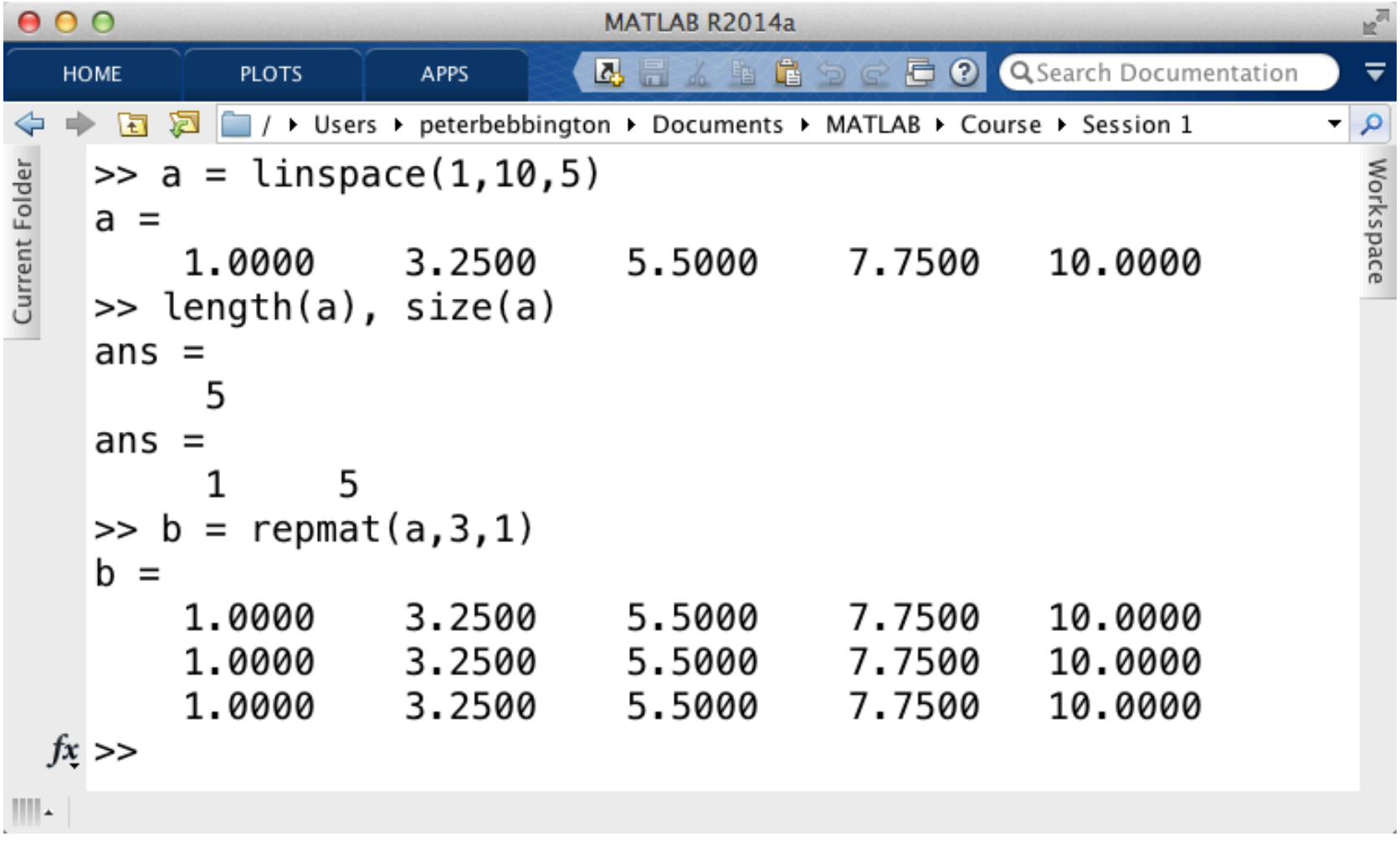
| Syntax | Operation |
|---|---|
| ’ | Transpose |
| + | Addition |
| − | Subtraction |
| * | Matrix Multiplication |
| .* | Elementwise Multiplication |
| $\wedge$ | Matrix Power |
| .$\wedge$ | Elementwise Power |

# SPECIAL MATRICES

- It is convenient in MATLAB to be able to build special matrices without the necessity of long procedures. The reason will become more evident as we progress through the MATLAB course.

| **Command** | **Meaning** |
|---|---|
| `ones(a,b)` | $a \times b$ matrix filled with ones |
| `zeros(a,b)` | $a \times b$ matrix filled with zeros |
| `eye(a)` | $a \times a$ identity matrix |
| `repmat(e,a,b)` | Replicates in $a \times b$ tiles the element $e$ |
| `rand(a,b)` | $a \times b$ random matrix (uniformly distributed) |
| `randn(a,b)` | $a \times b$ random matrix (normally distributed) |
| `linspace(s,e,nr)` | Creates a linearly spaced row vector |
| `logspace(s,e,nr)` | Creates a log spaced row vector |

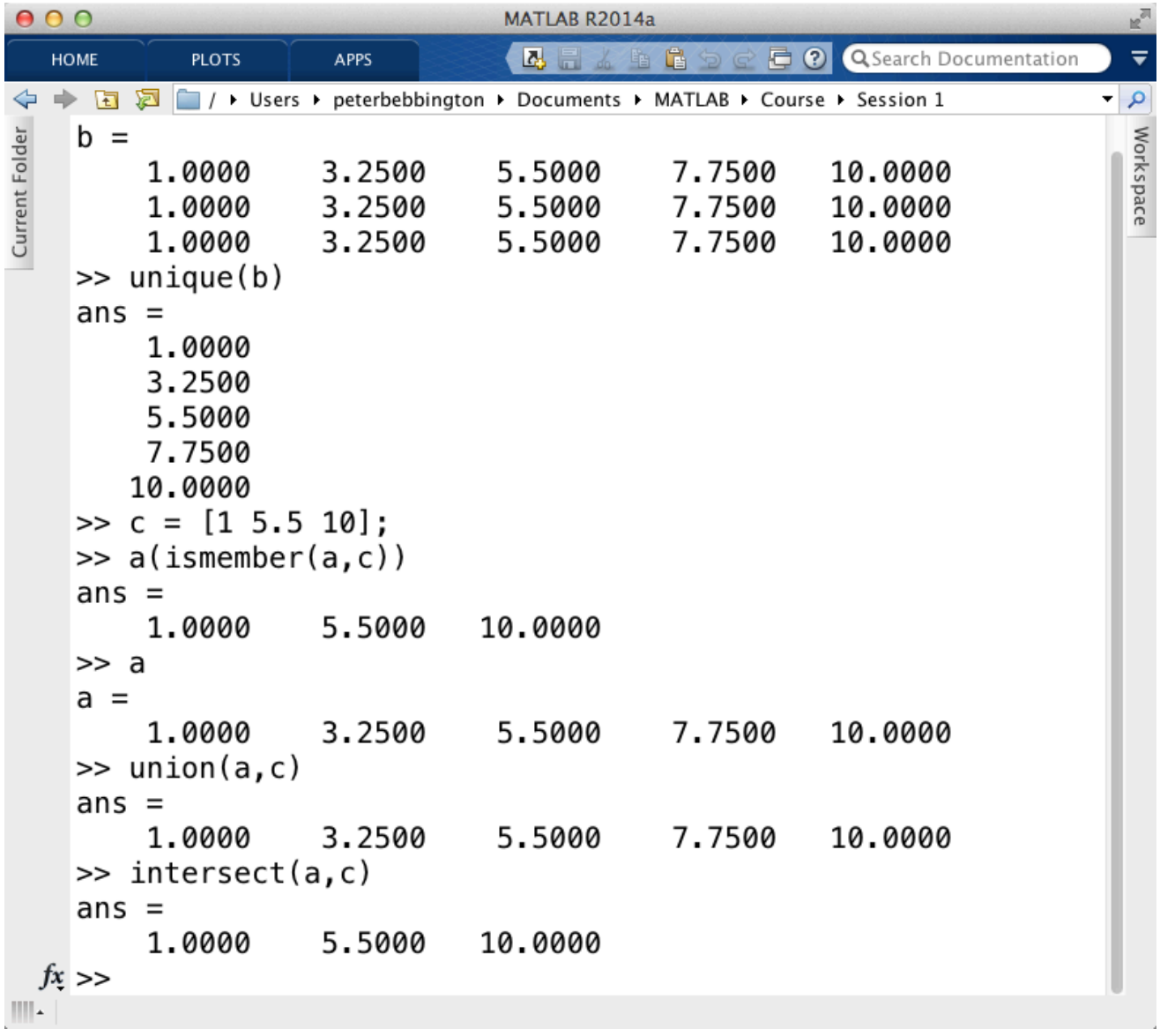
# SPECIAL MATRICES EXAMPLE



# SET FUNCTIONS

| Command | Meaning |
|---|---|
| `unique(a)` | Unique elements of a vector $a$ |
| `union(a,b)` | Union of sets $a$ and $b$ |
| `intersect(a,b)` | Intersection of sets $a$ and $b$ |
| `ismember(a,b)` | Boolean for $a$ elements in $b$ |
| `setdiff(a,b)` | Elements in $a$ that are not in $b$ |

# SET FUNCTIONS EXAMPLE



# SYSTEMS OF LINEAR EQUATIONS

- As MATLAB is an interpreted programming language it is intuitive to solve problems in Matrix form rather than using control flow with if/while/for loops. Consider the following example: system of equations.

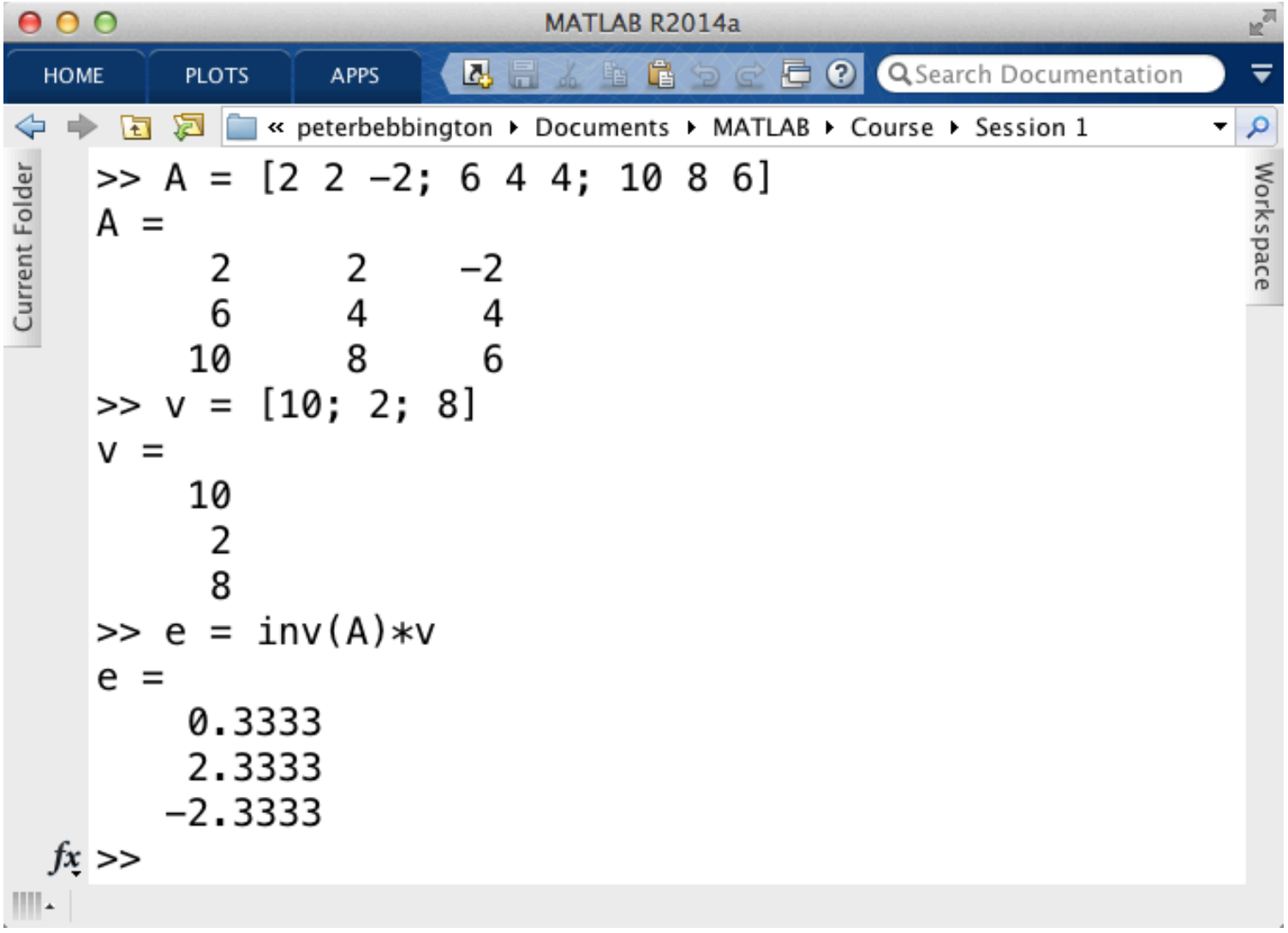
$$2x + 2y - 2z = 10$$
$$6x + 4y + 4z = 2$$
$$10x + 8y + 6z = 8$$

- These equations can be represented by

$$\bar{\bar{A}}\bar{e} = \bar{v}$$

# SOLVING LINEAR EQUATIONS

- Solved by inverting matrix $\bar{\bar{A}}$ and multiplying it by $\bar{v}$.

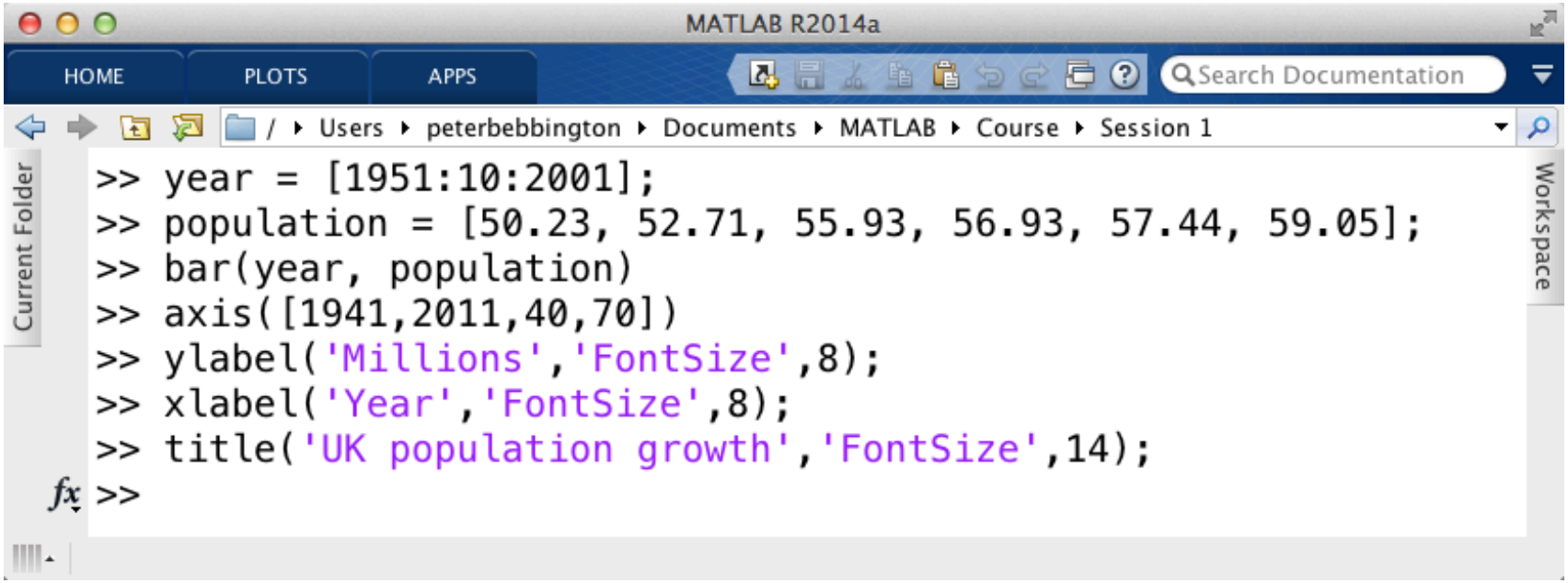


# PLOTTING FUNCTIONS

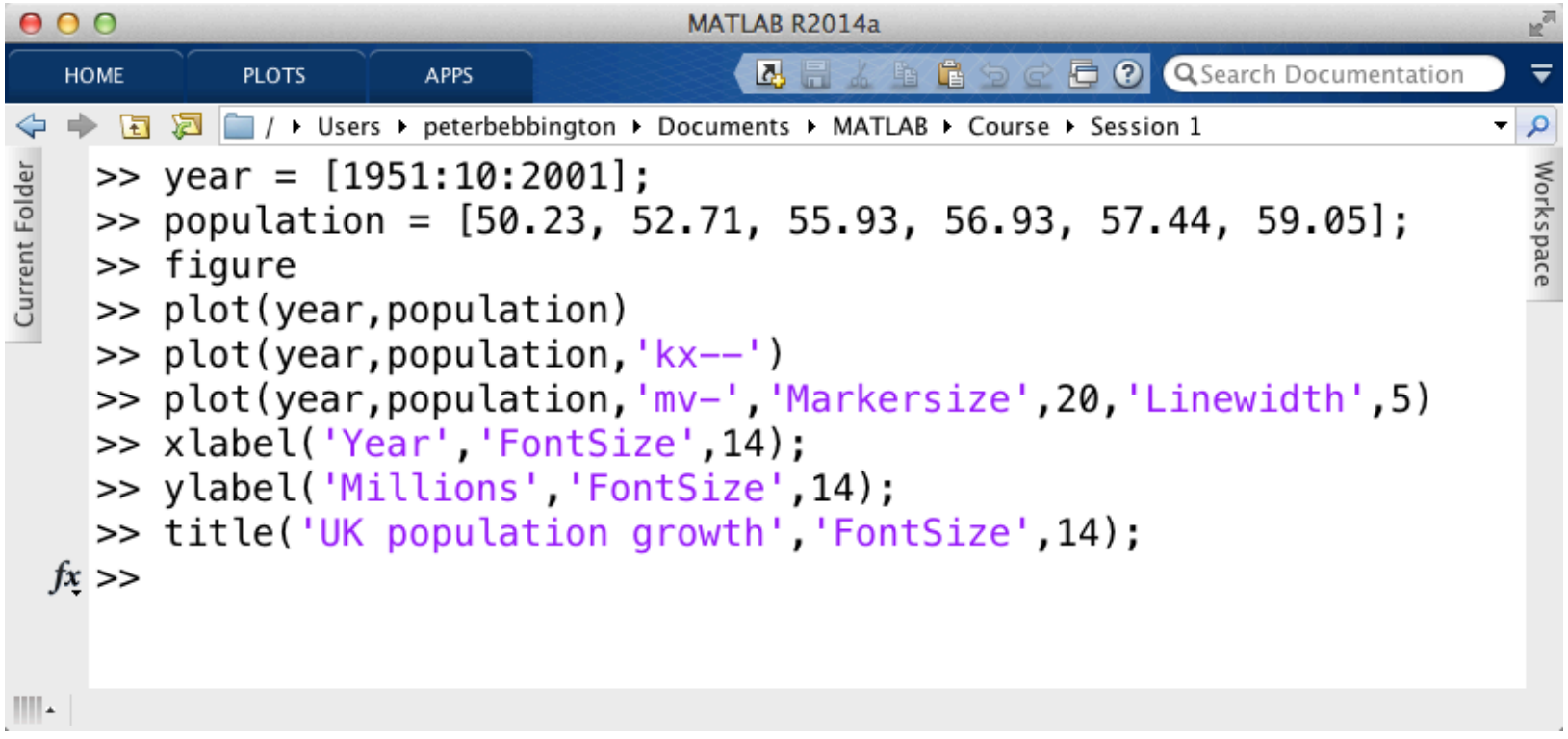
| Command | Meaning |
|---|---|
| `figure` | new figure window |
| `figure(n)` | n-th figure window |
| `title('string')` | add title to the window |
| `gtext('string')` | add text with the mouse |
| `legend('string')` | add legend |
| `grid on/off` | swith grid on/off |
| `subplot(m,n,p)` | multiple plots in one figure window |
| `hold on/off` | hold the current contents |

- Note that once a figure is created Matlab can auto generate code to reproduce the plot by `Figure -> File -> Generate Code`

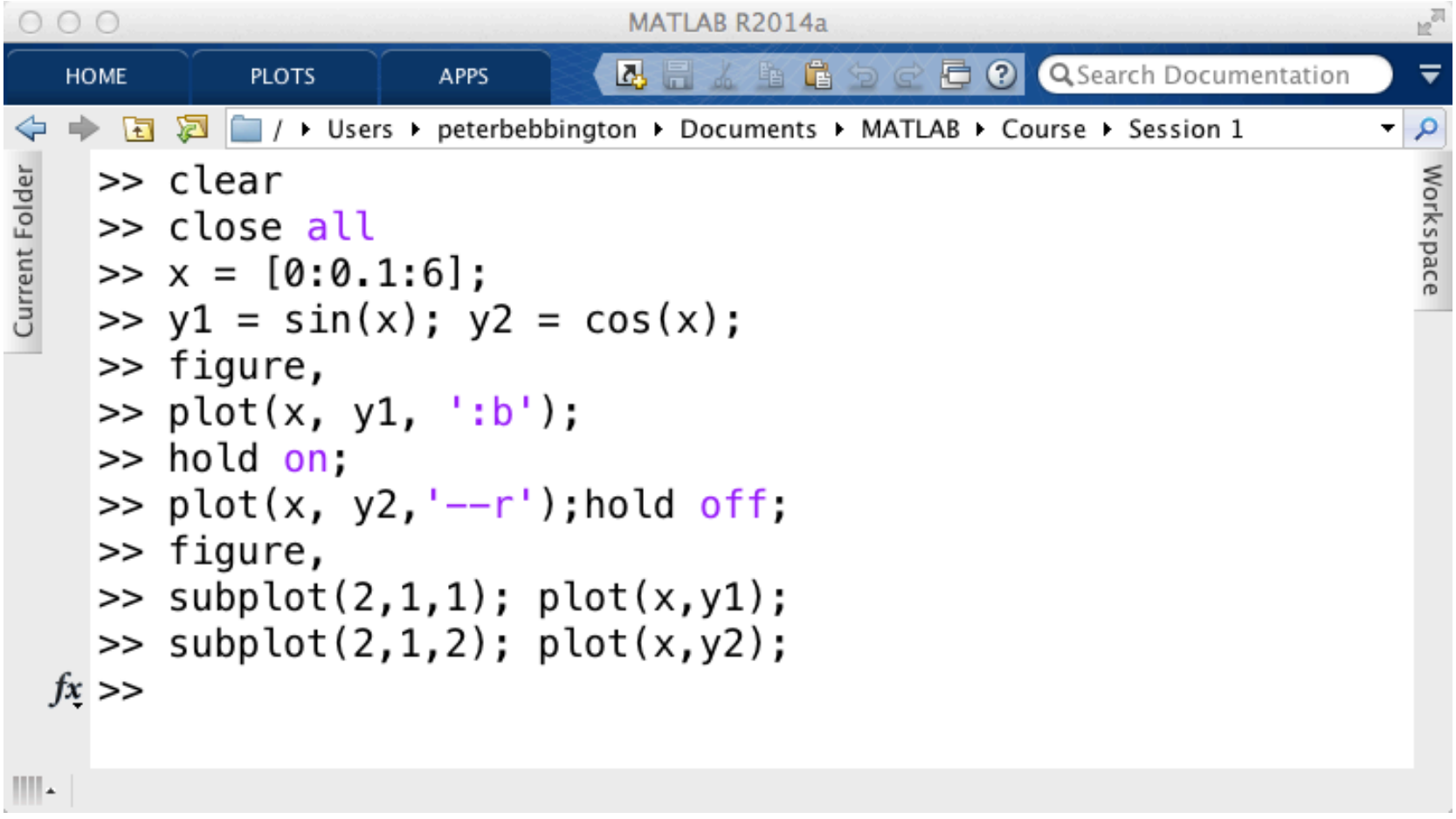
# BAR PLOTS


MATLAB R2014a
HOME
PLOTS
APPS
Search Documentation
/ ▸ Users ▸ peterbebbington ▸ Documents ▸ MATLAB ▸ Course ▸ Session 1
Current Folder
Workspace
>> year = [1951:10:2001];
>> population = [50.23, 52.71, 55.93, 56.93, 57.44, 59.05];
>> bar(year, population)
>> axis([1941,2011,40,70])
>> ylabel('Millions','FontSize',8);
>> xlabel('Year','FontSize',8);
>> title('UK population growth','FontSize',14);
>>
fx

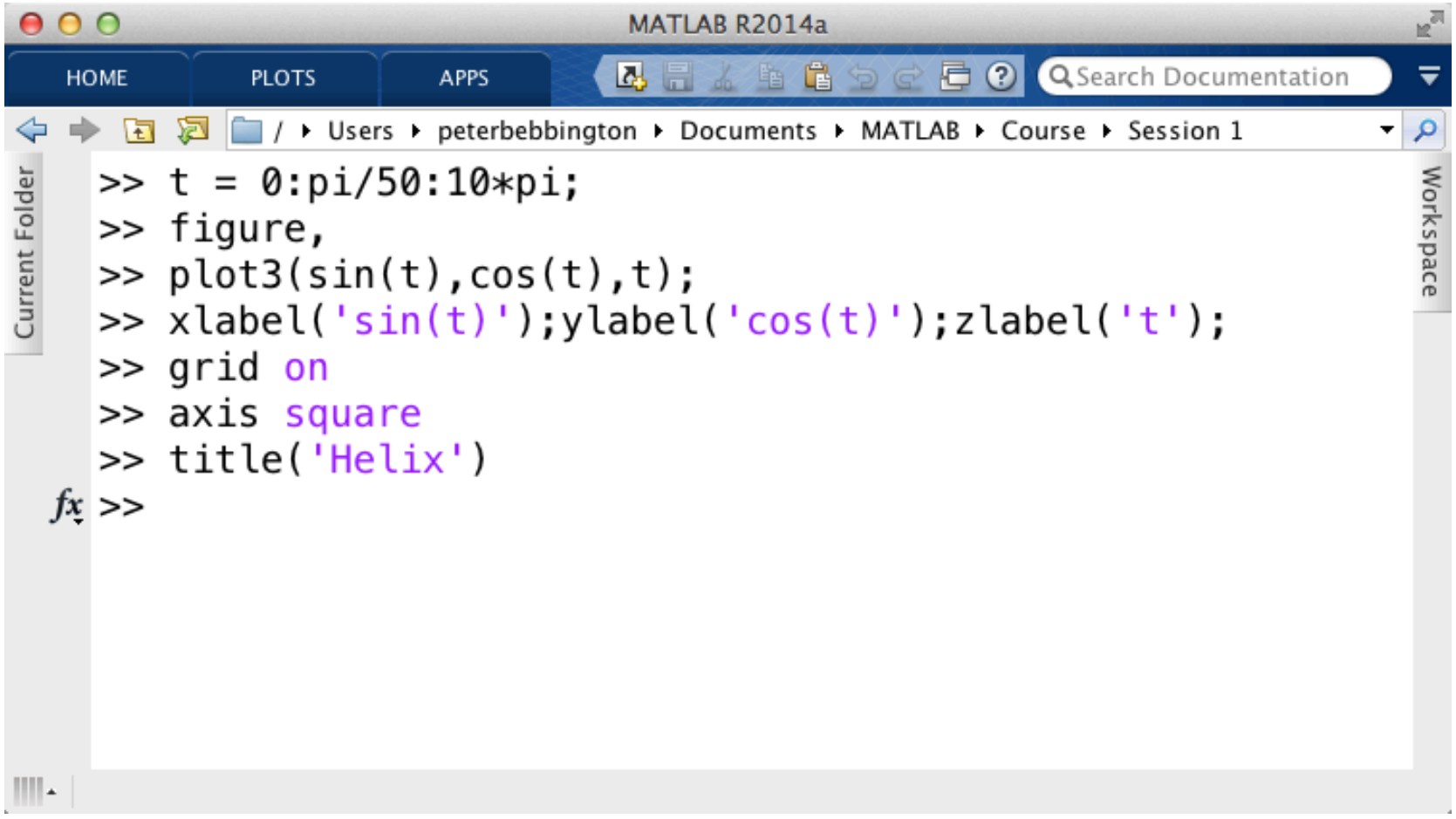
# LINE PLOTS


MATLAB R2014a
HOME
PLOTS
APPS
Search Documentation
/ ▸ Users ▸ peterbebbington ▸ Documents ▸ MATLAB ▸ Course ▸ Session 1
Current Folder
Workspace
>> year = [1951:10:2001];
>> population = [50.23, 52.71, 55.93, 56.93, 57.44, 59.05];
>> figure
>> plot(year,population)
>> plot(year,population,'kx--')
>> plot(year,population,'mv-','Markersize',20,'Linewidth',5)
>> xlabel('Year','FontSize',14);
>> ylabel('Millions','FontSize',14);
>> title('UK population growth','FontSize',14);
>>

# HOLD AND SUBPLOT

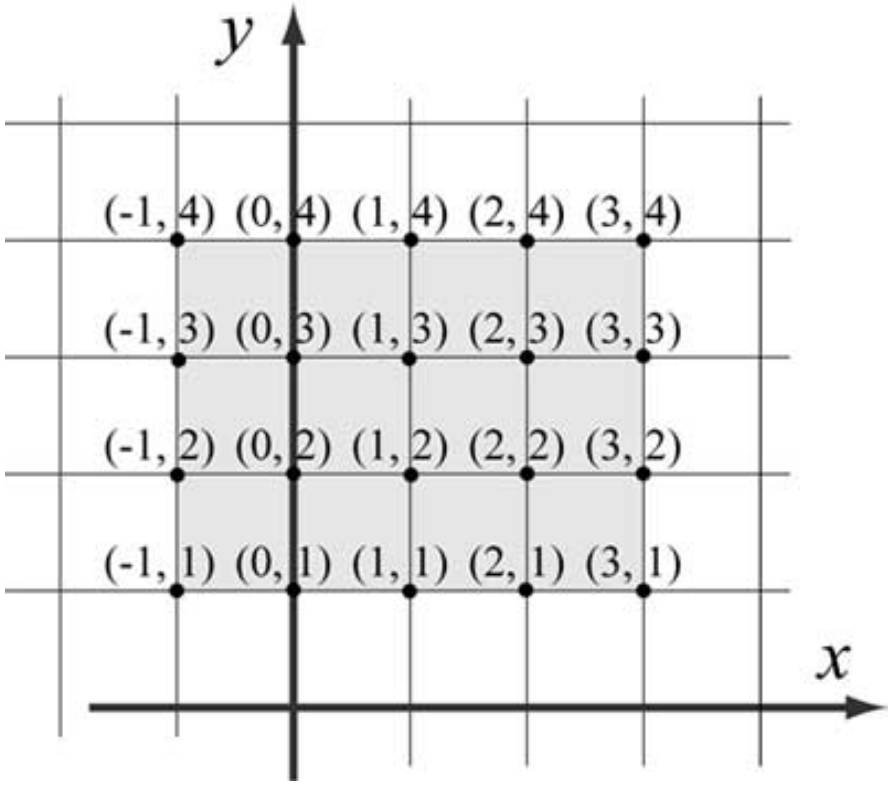


# 3D PLOTS


MATLAB R2014a
HOME
PLOTS
APPS
Search Documentation
/ ▸ Users ▸ peterbebbington ▸ Documents ▸ MATLAB ▸ Course ▸ Session 1
Current Folder
Workspace
>> t = 0:pi/50:10*pi;
>> figure,
>> plot3(sin(t),cos(t),t);
>> xlabel('sin(t)');ylabel('cos(t)');zlabel('t');
>> grid on
>> axis square
>> title('Helix')
>>

# MESH, SURFACE, CONTOUR

- Meshgrid creates the matrices of $x$ and $y$ co-ordinates
- mesh creates mesh plots
- surf creates surface plots
- contour creates contour plots.



# MESH, SURFACE, CONTOUR PLOTS