MATLAB for Finance Course: Session 1

Dr. Peter A. Bebbington

Brainpool AI

peter@brainpool.ai

bebbington @brainpoolai @brainpoolai

November 16, 2024

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. Consultancy brainpool.ai
- 2021-Present: AI Director for daisy.ai
- 2024-Present: CTO for tunedd.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 ENVIRONMENT

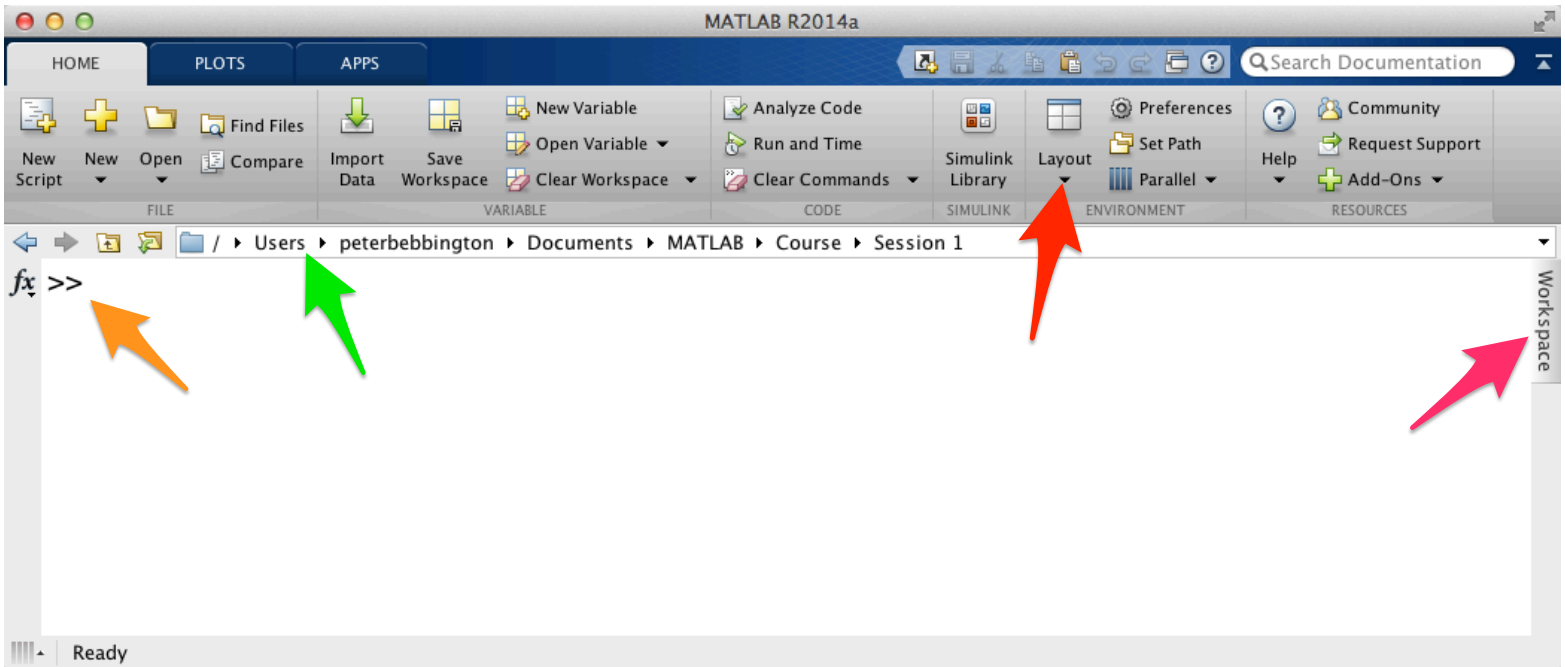
- MATLAB provides an integrated development environment (IDE)
- Key components:
  - Editor - for writing scripts and functions
  - Debugger - for finding and fixing errors
  - Profiler - for optimizing code performance
  - Version Control Integration
- Built-in math functions and libraries
- Extensive toolbox ecosystem
- Interactive documentation and examples

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
  - ChatGPT or/and Claude

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.
- 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.

```
%% Command Window
why

%% Numerical Commands
-11/3, pi^2, date, 2+3;
```

VARIABLES

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

```
%% Variables
(-1)^5;
ans
newvariable = ans^2
```

VARIABLE TYPES

- Numeric Types:
  - `double` (default)
  - `single`
  - integer types (`int8, uint8, etc.`)
- Character and String Types:
  - `char`
  - `string` (newer type)
- Logical Type:
  - `true/false` (1/0)
- Complex Numbers:
  - Using i or j as imaginary unit
  - Example: `3 + 4i`

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 Example

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

```
%% Arrays
x = [1,2,3,4], y = x*2
x = 1:5; y = x + 1.1
x = 0:10:50; y = x * 0.1

%% Building Matrices
A = [1,2,3; 5,6,7]
B = [1:0.5:3]
M = magic(5) % Magic square
```

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 Operations

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

```
%% Concatenating
A = [1,2;3,4];
B = [3,4;1,2];
C = [A;B]

%% Matrix Elements
a = 1:16;
D = reshape(a,4,4)
D(9)
D(3,1)
```

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

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

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

Special Functions

```
%% Special Matrices
a = linspace(1,10,5)
length(a), size(a)
b = repmat(a,3,1)

%% Set Functions
unique(b)
c = [1 5.5 10];
a(ismember(a,c))
union(a,c)
intersect(a,c)
```

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

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

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

VISUALIZATION BASICS

- Plot Types:
  - 2D plots (plot, scatter, bar)
  - 3D plots (plot3, surf, mesh)
  - Specialized plots (histogram, boxplot)
- Figure Properties:
  - Axes labels and titles
  - Line styles and colors
  - Markers and annotations
  - Multiple subplots
- Export Options:
  - Save as image (png, jpg)
  - Save as vector graphics (pdf, eps)
  - Generate MATLAB code

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`

Plotting Examples

```
%% Basic Plot
year = [1951:10:2001];
population = [50.23, 52.71, 55.93, 56.93, 57.44,
    59.05];
plot(year,population,'kx--')
ylabel('Millions','FontSize',14)
xlabel('Year','FontSize',14)
title('UK population growth','FontSize',14)
```

Bar Plot Example

```
%% Bar Plot
figure
bar(year, population, 'FaceColor', [0.2 0.6 0.8])
hold on
plot(year, population, 'r*', 'MarkerSize', 10)
ylabel('Millions','FontSize',14)
xlabel('Year','FontSize',14)
title('UK population growth (Bar Chart)','FontSize'
    ,14)
grid on
```

- Combines bar chart with point markers
- `FaceColor` sets the color of the bars
- `MarkerSize` controls the size of the points

Hold and Subplot Example

```
%% Hold and Subplot
clear
close all
x = [0:0.1:6];
y1 = sin(x); y2 = cos(x);
% Plot with hold on
figure,
plot(x, y1, ':b');
hold on;
plot(x, y2,'--r');
hold off;
% Plot with subplots
figure,
subplot(2,1,1); plot(x,y1);
subplot(2,1,2); plot(x,y2);
```

- `hold on` allows multiple plots in same figure
- `subplot(m,n,p)` creates grid of plots: m rows, n columns, p position

Scatter Plot Example

```
%% Scatter Plot with Size and Color Variation
n = 50;
x = randn(n,1);
y = randn(n,1);
sizes = randi([20,100], n,1);
colors = rand(n,1);
figure
scatter(x, y, sizes, colors, 'filled')
colorbar
title('Scatter Plot with Variable Size and Color')
xlabel('X Values'), ylabel('Y Values')
```

- Scatter plot shows how to vary marker size and color
- Colors can represent a third dimension of data
- Size variations highlight data importance

Histogram Example

```
%% Histogram with Customization
figure
data = randn(1000,1);
histogram(data, 30, 'FaceColor', [0.4 0.6 0.8], ...
    'EdgeColor', 'black', 'FaceAlpha', 0.7)
title('Histogram of Normal Distribution')
xlabel('Values'), ylabel('Frequency')
grid on
```

- Histogram demonstrates distribution visualization
- Customizable appearance with various properties
- Useful for analyzing data distributions

Multiple Plot Types Example

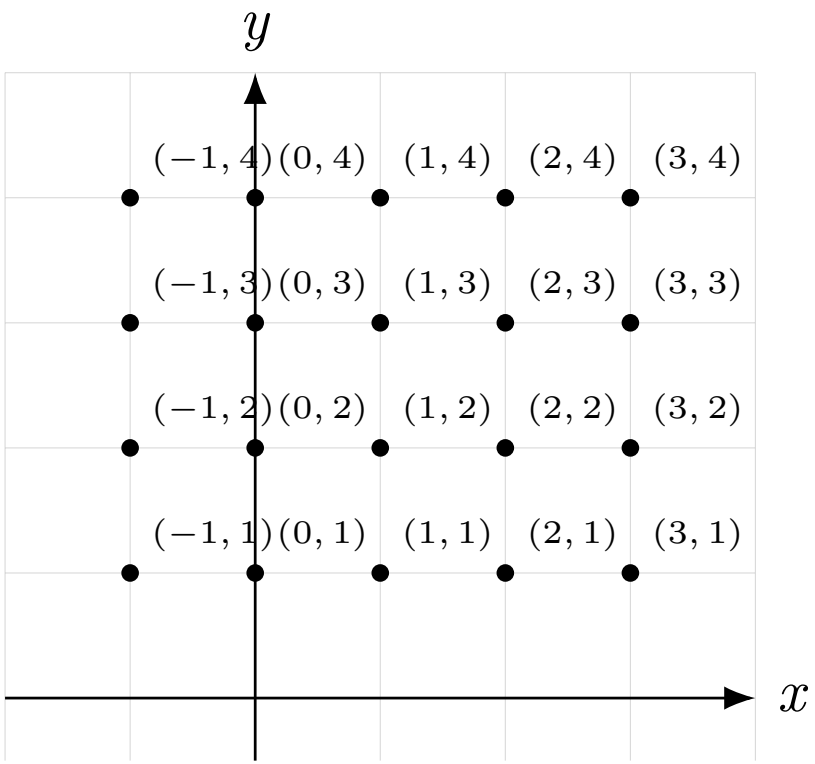
```
%% Combining Different Plot Types
x = linspace(0, 2*pi, 100), y1 = sin(x), y2 = cos(x
    );
figure
subplot(2,2,1)
plot(x, y1, 'b-', 'LineWidth', 2)
title('Line Plot')
grid on
subplot(2,2,2)
stem(x(1:10:end), y1(1:10:end), 'r')
title('Stem Plot')
grid on
subplot(2,2,3)
stairs(x(1:10:end), y2(1:10:end), 'g')
title('Stairs Plot')
grid on
subplot(2,2,4)
area(x(1:10:end), y1(1:10:end))
title('Area Plot')
grid on
```

Advanced Plotting

```
%% 3D Plot Example
t = 0:pi/50:10*pi;
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, and Contour Plots

```
%% Mesh, Surface, Contour Plots
x = [0:0.1:4]; y = [0:0.1:4];
[X,Y] = meshgrid(x,y);
Z = 2*sin(X) + cos(Y);

% Create mesh plot
figure,
mesh(X,Y,Z)

% Create surface plot
figure,
surf(X,Y,Z)

% Create contour plot
figure,
contour(X,Y,Z)
```

Note: These commands create 3D visualizations of the function $Z = 2\sin(X) + \cos(Y)$

EXERCISERS

A. Create the following matrices in Matlab:

$$\bar{\bar{A}} = \begin{pmatrix} 5 & 3 & 1 \\ 3 & 4 & 7 \\ 1 & 7 & 5 \end{pmatrix} \quad \bar{b} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} \quad \bar{\bar{C}} = \begin{pmatrix} 6 & 6 & 0 & 2 \\ 9 & 2 & 1 & 3 \end{pmatrix}$$

1. Combine matrices $\bar{\bar{A}}$, $\bar{\bar{B}}$ and $\bar{\bar{C}}$ in all possible ways (horizontally and vertically) using the constructor $[\,]$ and the operators ; (Semicolon) and , (Comma).
2. Solve the linear problem $\bar{\bar{A}}\bar{e} = \bar{b}$ by finding eigenvector $\bar{e}$ and eigenvalues.

B. Using the : (Colon) create the following arrays:

```
a = 1 2 3 4 5 6
b = 2 2.5 3 3.5 4 4.5 5
c = 3 2.75 2.5 2.25 2 1.75 1.5 1.25 1
```

1. Write code that will add the 1st, 3rd and 6th element of the array a. This sum should should be placed as an extra element at the end of the array. Do the same for arrays b and c.