MA1710: Week 1

Introduction to Matlab.

Every modern mathematician, especially every mathematician who wishes to apply maths to the solution of real-life problems must be computer-literate. This series of labs is designed to give you a brief introduction to Matlab — one of the most popular computational environments for mathematicians. Computing things using Matlab, as a powerful calculator and/or as a programming language, will constitute an important part of your degree.

Before we begin the session, a brief note about the format of these handouts is in order. The handouts are designed to be used both during the lab sessions and also after them, for independent self-study. The teaching during the lab sessions will be mainly focussed on topics that are typeset using the regular font. Additional topics, which are indicated by the smaller font, may or may not be covered during the labs (this depends on how quickly the class will progress with the material in every session), but are still very useful for you to know. All handouts also have additional notes/examples/exercises at the end, which are designed to aid your revision and, sometimes, to expose you to further materials for independent learning.

1.1 Customizing your set-up

To make things a little easier in how Matlab starts, where you save your files and for us to more quickly help you in some cases please can you do the following.

1. After you have logged in start internet explorer and open the file

   `http://people.brunel.ac.uk/~icstmkw/ma1710/lab_setup.html`

2. Keep the web browser open and get a command prompt window by clicking on Start (bottom left hand corner of the screen) and typing `cmd.exe` in the box shown.

3. Highlight the 3 lines given below in the web browser and press together the `Ctrl` and `C` keys to save these to a buffer. Next make the `cmd.exe` window your active window, select `Edit` from the top bar and from the sub-menu select `Paste`. This should execute the batch of lines that you have highlighted. The 3 lines are as follows.

   ```
   net use r: /d
   net use r: \\uxisapp1.brunel.ac.uk\apps
   r:\c++\ma1710\u\mybin15\mkw_mat15.bat
   ```

   You should not need to do this again.

1.2 The Matlab environment

Matlab is available in every computer lab that you are likely to use at Brunel and you should be able to use it on every networked PC in the university although the way to start it may vary a little. Assuming that you are in the Halsbury building (`HALB`) or the the Saint Johns building (`STJN`) or the John Crank building (`JNCK`) then all the computers currently run Windows 7 Enterprise and the easiest way to start Matlab is to click the Start button (bottom left hand corner), type `matlab` into the search field, and click the link MATLAB (all-capital letters), see Figure 1.1 on page 3.

The university supports several versions of Matlab. Usually, you will use this version of Matlab (the one we started, i.e. MATLAB). However, on some rare occasions, it may so happen that all licenses for this version of the Matlab are currently being used by someone else. In this case you are advised to start any other version of Matlab near the top of the list. You will not have any difficulties adopting to a slightly different version because the differences between versions are relatively minor.

Once Matlab is loaded, you will be presented with the main Matlab window, containing several panels, see Figure 1.4 (note that we numbered the most important items in the Matlab window). The panel that we need to talk about first is the panel with the number 1 — the *Command Window*.

Matlab offers great flexibility in customizing its layout. Each of the panels can be resized, moved around and even detached from the main window. There are also several additional panels that are not shown by default, but can be added to the main window. Unfortunately, this flexibility means that it is sometimes easy to close one of the important panels or to rearrange panels in a way that is not convenient for work. This is why sometimes you will want to reset the panel settings to their defaults. With our versions of Matlab this can be done by selecting the menu item `Desktop`→`Desktop layout`→`Default`.

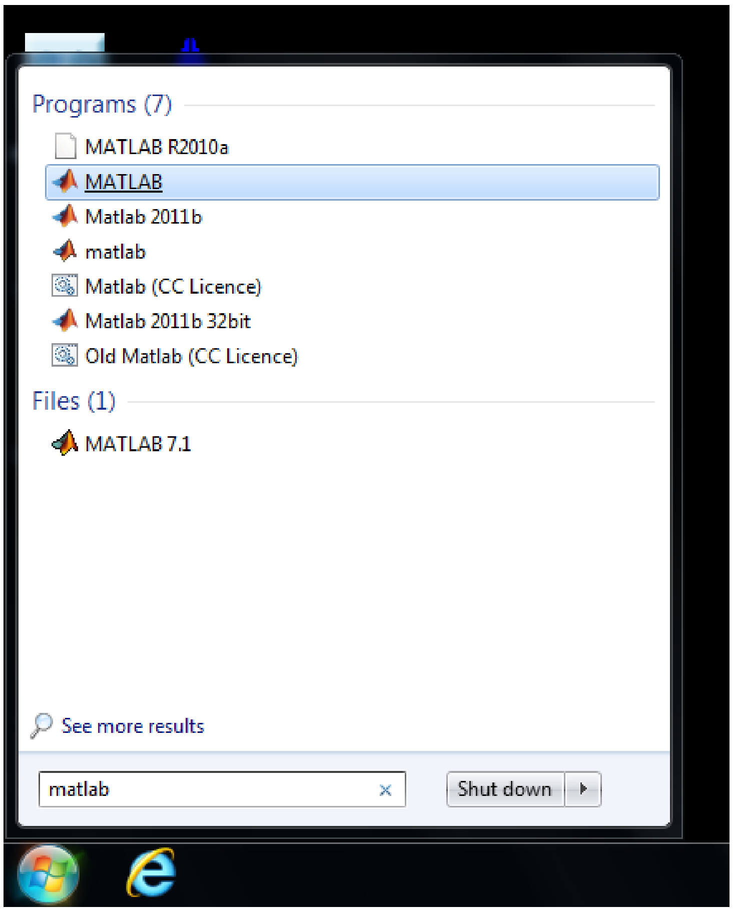
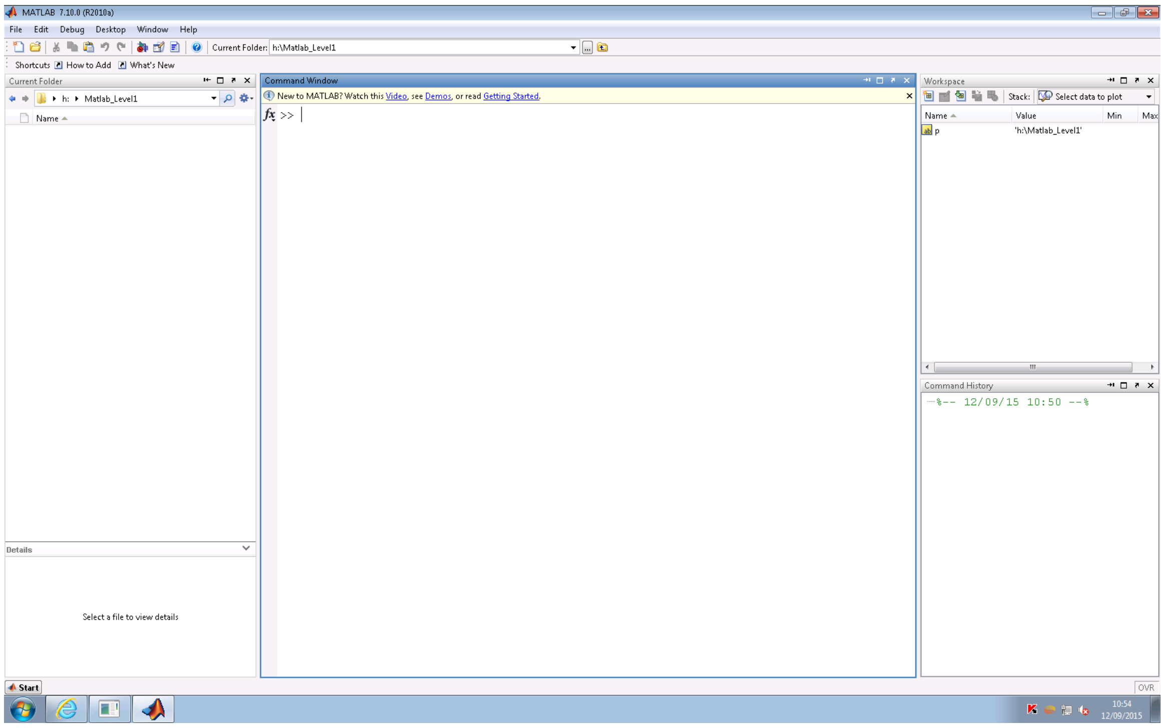


Figure 1.1: Starting Matlab


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Current Folder: h:\Matlab_Level1
Shortcuts How to Add What's New
Current Folder
Command Window
Workspace
Command History
Details
Start
OVR

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

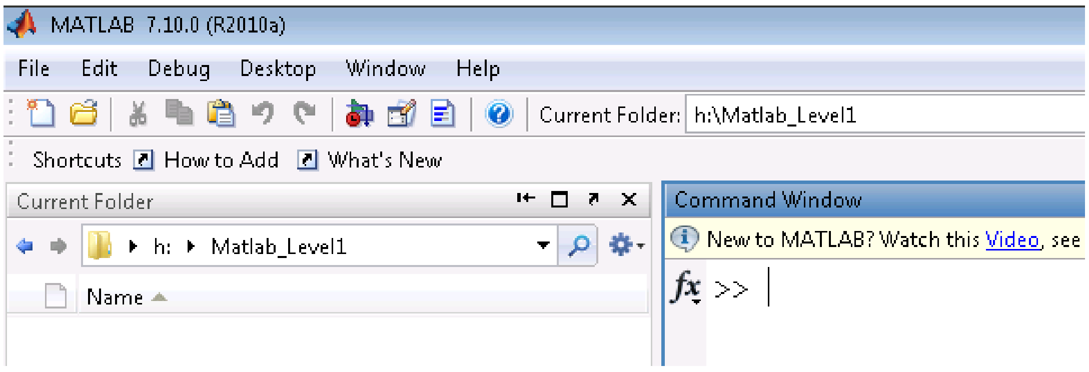
```
fx >> |
```

Name	Value	Min	Max
p	'h:\Matlab_Level1'		

```
%-- 12/09/15 10:50 --%
```

Select a file to view details

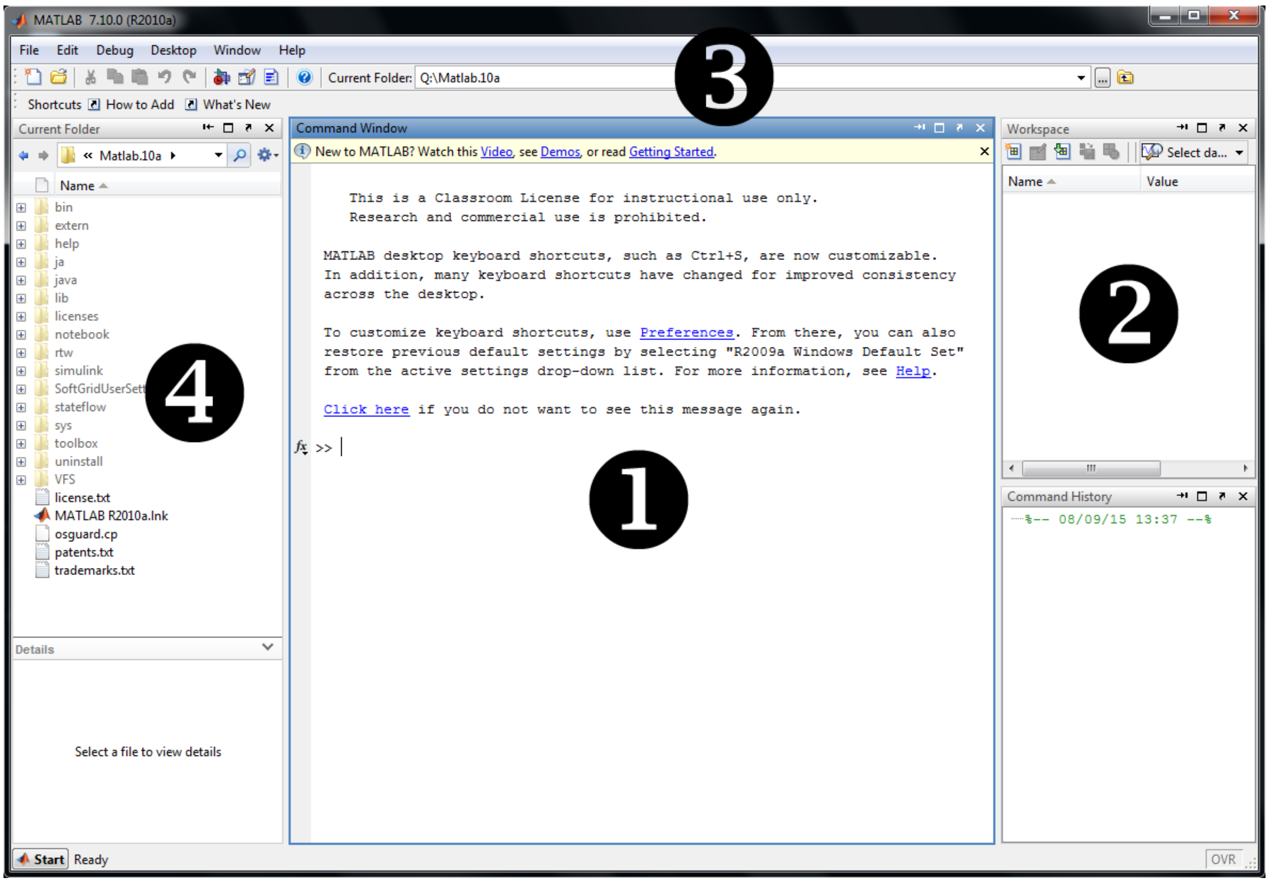
Figure 1.2: The layout of the main Matlab window as a result of the set-up instructions,


MATLAB 7.10.0 (R2010a)
File Edit Debug Desktop Window Help
Current Folder: h:\Matlab_Level1
Shortcuts How to Add What's New
Current Folder
h: Matlab_Level1
Name
Command Window

New to MATLAB? Watch this Video, see

```
fx >> |
```

Figure 1.3: The top left hand corner of the above window.



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

MATLAB desktop keyboard shortcuts, such as Ctrl+S, are now customizable.
In addition, many keyboard shortcuts have changed for improved consistency
across the desktop.

To customize keyboard shortcuts, use Preferences. From there, you can also
restore previous default settings by selecting "R2009a Windows Default Set"
from the active settings drop-down list. For more information, see Help.

Click here if you do not want to see this message again.

>> |
```

```
%-- 08/09/15 13:37 --%
```

Figure 1.4: The default layout of the main Matlab window if you did not do the set-up.

1.3 Matlab as a calculator

The first thing you need to find in the Command Window is the Matlab command prompt:

```
>>
```

This is where you can input commands for Matlab to execute.

The simplest kind of commands for Matlab are mathematical expressions, so, at first, you can think of Matlab Command Window as a powerful interactive calculator. You can type in any mathematical expression you like, press "Enter" button and Matlab will do its best to compute it. The four basic mathematical operations — addition, subtraction, multiplication and division — are denoted in Matlab by symbols `+`, `-`, `*` and `/`. The symbol `^` denotes raising to power.

Exercise 1.3.1

Try typing each of the following expressions into the Command Window prompt.

```
2+2
111-12
143*7
24/3
5^2
```

Have you obtained the answers that you expected?

Answer of exercise 1.3.1

What should appear in your command window is the following.

```
ans =
     4
ans =
    99
ans =
          1001
ans =
     8
ans =
    25
```

You learned during your GCSEs that the order in which any mathematical expression must be evaluated is not arbitrary and, in fact, follows some well-defined rules (BIDMAS, anyone?). The letters stand for brackets, indices, division, multiplication, addition and subtraction. There are also slight variations as to the acronym used but the rules are the same. Matlab, just like any almost every other modern computing environment/programming language implements a generalized version of these rules by assuming the following *levels of precedence*:

1. Parentheses `()`, innermost first
2. Powers (^)
3. Unary plus (+), unary minus (-)
4. Multiplication (*), division (/)
5. Addition (+), subtraction (-)

When Matlab needs to compute an expression, it first evaluates operators at the highest level of precedence, then operators at the second level of precedence, then third, etc. Within each precedence level, operators are computed from left to right.

Exercise 1.3.2

For each of the following expressions, first compute the value yourself, then type the expression into the Matlab Command Window to see if your answers match the answers given by Matlab:

```
2+2*2
3*2^2
4*-1-2
12/4*3
-1-2-4*-3/2
```

Answer of exercise 1.3.2

There are no syntax error in any of the above and the output generated is as follows.

```
ans =
     6
ans =
    12
ans =
    -6
ans =
     9
ans =
     3
```

Note that there are 5 commands and 5 numbers displayed.

How many of your answers matched the Matlab outputs? Please think about each of the cases where you got it wrong and try to apply the rules of precedence, as literally as you can, to see where you made a mistake. Whether you made a mistake or not it would be wise not to actually write the expressions as given but instead to add optional brackets so that it is clear without having to study the expression too closely. A better way of writing the expressions to get the same answers is to write the following.

```
2+(2*2)
3*(2^2)
4*(-1)-2
(12/4)*3
-1-2-4*(-3)/2
```

Exercise 1.3.3

Compute the following values using Matlab:

$$\text{(a)}\ \frac{36}{3\cdot 6}\,, \qquad \text{(b)}\ 3^2\frac{1+2}{4+5}-1\,, \qquad \text{(c)}\ 1+\frac{3}{2}-0.5\,.$$

(Note that in part (a) the denominator does mean 3 times 6.) Use brackets in every case where the default order of evaluation is not going to work.
The correct answers are: (a) 2, (b) 2 and (c) 2.

Answer of exercise 1.3.3

One possible answer is to write the following.

```
36/(3*6)
(3^2)*(1+2)/(4+5)-1
1+(3/2)-0.5
```

Some of the round brackets are not needed but they possibly slightly clearer than if they are not used.

Part (c) of Exercise 1.3.3 features a floating point number. You can type it into Matlab as is, `0.5`, and Matlab will interpret it correctly. All of our exercises thus far were designed to produce integer answers; however, it does not have to be the case. By default, Matlab performs all its computations using double accuracy floating point numbers, which means that you can usually get approximately 15–16 significant digits in most of your computations.

By default, Matlab will not be printing all significant digits for your floating point results, because it is usually not needed. In fact, you can specify the precision of the Command Window outputs that you get by default by going to the menu item `File`→`Preferences`→`Command Window` and selecting `Numeric format` of your choice (`short` or `long` are good default choices). You may also want to select `Numeric display`→`Compact` in the same window, so that the Command Window can fit more results of your computations. All of the above can also be achieved by using the command window by typing, for example, the following.

```
format compact
format long
```

These change to the compact mode (no blank lines between outputs) and the mode when about 15 decimal digits are displayed for each number. You replace `long` by `short` to get to the mode when only about 5 or 6 digits are shown.

Floating point numbers may often be written more concisely using the scientific notation, in which a number is represented as a product of a number and a power of ten. For example,

$$1000 = 1 \times 10^3 = \texttt{1E3}, \qquad 5 \text{ million} = 5 \times 10^6 = \texttt{5e6},$$
$$\frac{5.2}{1000000} = 5.2 \times 10^{-6} = \texttt{5.2E-6}$$

As you can see, instead of writing the factor 10^n explicitly, one writes the letter `e` or `E` followed by the power needed (the letter `e` or `E` stands for "exponent"). Matlab will automatically switch to the scientific notation whenever it has to output numbers that are relatively small or relatively large.

Exercise 1.3.4

Write in scientific notation the answers to the the following expressions. You can check your answers using division as indicated.

```
1/10
1/100
1/1000000
```

Whenever you need to type several similar expressions into the Command Window, you can benefit from the history of past commands stored for you by Matlab. Hence, after computing `1/10`, press the "Up" button to obtain the previous input (`1/10`) in your current command prompt. You can press "Up" or "Down" buttons as many times as necessary to browse through all your past Command Window inputs. There is also a panel on the right hand side in the default layout which shows the command history.

Answer of exercise 1.3.4

To use scientific notation in all cases you can have the following.

```
1e-1
1e-2
1e-6
```

In the first two cases can of course easily be written as `0.1` and `0.01` respectively. The point about this exercise is that you can just write the answer down avoiding and division operation.

In addition to the standard algebraic operations, Matlab makes available a huge number of mathematical functions. Functions are computed by typing the function name followed by the function argument in round brackets. For example, square roots are computed using function `sqrt`, so to compute $\sqrt{2}$ one needs to type into the Command Windows prompt:

```
sqrt(2)
```

Generally speaking, most function names are chosen to be as close as possible to the standard names of mathematical functions, so on many occasions you can simply guess the correct name.

Exercise 1.3.5

Compute the following expressions:

```
pi
sin(0)
cos(pi)
tan(pi/4)
exp(1)
log(1)
abs(-33)
```

Note that Matlab trig-functions does not compute degrees, only radians. Matlab actually has many functions and in particular it has a version with a different name which does work with degrees. Try for example the following.

```
sind(0)
cosd(180)
tand(45)
```

Answer of exercise 1.3.5

With the format being in the `long` mode the output from the the first set of commands is as follows.

```
ans =
   3.141592653589793
ans =
     0
ans =
    -1
ans =
   1.000000000000000
ans =
   2.718281828459046
ans =
     0
ans =
    33
```

The output from the second version using `sind` and `tand` is as follows.

```
ans =
     0
ans =
    -1
ans =
     1
```

The difference in the result shown for `tan(pi/4)` and `tand(45)` is as a consequence of rounding error.

`abs` is likely to be the least intuitive function name in this exercise. It stands for the absolute value of a number. Whenever in doubt, just type `help` followed by the name of the function, or type `doc` followed by the name of the function, that you find unclear, e.g.

```
help abs
doc abs
```

Even more detailed help, often with examples, can be obtained by single-clicking *inside* of the word `abs` in the Command Window and pressing button `F1`. Such detailed help is available in Matlab for every single function.

You typed quite a few things into the Command Window and it is getting a little untidy. Matlab command `clc` clears the Command Window. I would recommend that you use this sparingly and only when you are sure you no longer wish to see things previously displayed in the window.

1.4 Using variables

All expressions we considered thus far were extremely simple. When one has to deal with more complex expressions, it becomes extremely convenient to introduce and, sometimes, re-use intermediate results of your calculations. Most calculators come with one or several memory cells for this purpose. Matlab comes with the ability to introduce and use *variables*.

The idea of a variable can be somewhat unintuitive, so it is useful to try and suggest a metaphor for what a variable is. The key to this is in remembering that a variable is *a unit of storage*. The best way to imagine a variable is to think of it as if it was a named box with a number inside. We can choose a name for it and we can *assign a value to it,* in other words, store the value in our box.

The name of a variable must begin with a letter or underscore (`_`), but otherwise can be `almost` an arbitrary combination of letters, digits and underscores. You cannot have spaces, any of the brackets or any of the characters that you for arithmetic operations such as `+`, `-` etc.. The case of letters matters, i.e. names `x` and `X` would correspond to two different variables. We say that the combination is almost arbitrary, because you cannot use the name of existing Matlab command or function as a name for a variable (it would be very confusing anyway). So, just as an example, type into the Command Window

```
x=5
```

Two things will happen. First, Matlab will output a response into the Command Window, saying that x=5. Second, a new entry will appear in the Workspace window, see Figure 1.2, panel with the number 5. The new entry will also say that $x = 5$. The Workspace window, when there are not too many items, will be useful in that it will show you the present values for all currently defined variables. As another example, let us create a variable eulerE which stores the value of the Euler constant ($\mathrm{e} = \exp(1) \approx 2.71$). This can be done as

```
eulerE=exp(1)
```

Once variables have been defined, we can use them in future expressions. For example, we can input

```
2*x
eulerE^2
```

to obtain 10 and the value of e^2. In other words, if you use the name of a variable in an expression, during the computation it will be automatically replaced by the value of the variable.

The equals sign that we used to store values in variables x and eulerE must not be understood in the sense in which we understand equality in mathematics. The meaning of this operation, usually called *assignment*, is very particular: a value computed on the right-hand side of the equals sign is stored in the variable indicated on the left-hand side. Unlike in mathematics, you cannot write

```
x+y = 23
```

because x+y is not a variable (it is not a valid name for a variable) into which something can be stored. At the same time, you can *modify* values of variables, which leads to commands that may seem meaningless mathematically:

```
x = 2 * x
x = x + 1
```

(It is optional here whether or not extra spaces are put around some of the symbols.) Assuming that the original value of x was 5, the first of these commands will make x equal to 10, and the second command will further increment x by 1, so after executing these two commands the Workspace window will show that x is now equal to 11. To summarize here, the equal sign = means an assignment operation which is not the same as the equal sign that you use in an equation.

Exercise 1.4.1

What are the values of `a` and `b` after executing the following commands?

```
a = 10
b = 20
a = b
```

Predict the results and then check yourself by entering these commands into the Command Window and then referring to the Workspace panel.

Answer of exercise 1.4.1

The output in the command window is as follows.

```
a =
    10
b =
    20
a =
    20
```

The last statement ensures that once the statements have been executed both `a` and `b` store the same value.

Sometimes, after working on a larger problem, you may want to delete from Matlab memory the variables that you created. This can be done by using the command `clear`. Clear the memory and check that the Workspace panel is now empty, just like it was at the start.

1.5 Matlab scripts

Using the Command Window is good when you need to perform small, interactive computations. However, it limits the complexity of problems that you can solve and also leaves no record of your work, and thus if you need the results again you need to repeat again the commands that you used. Hence, even though we just spent a significant part of the session working with the Command Window prompt, this will not be the usual way to use Matlab in the future. Instead, we will usually write Matlab scripts for the type of work already described.

A Matlab script is a plain text file with the extension `.m` and is an example of what Matlab refers to as a m-file.

There are several ways that you can create and run Matlab script files and indeed you can use any editor you like to create the files. For these notes for this session we just mention some of the ways that you can do things.

If you were successful in the set-up instructions in section 1.1 then you are immediately ready to start. However, as a quick check and to partly appreciate what the set-up has done, observe what is given in the current folder bar near the top of the screen and also observe what is shown at the top of the current folder panel (see Figure 1.3). Both indicate what is the current folder and to confirm this further type `pwd` at the command prompt. Unless you have changed things the output should be as follows.

```
h:\Matlab_Level1
```

The set-up instructions created this folder and also put a file called `startup.m` in an appropriate location so that when Matlab starts the current folder is changed to this location. The part `h:` in the name indicates that it is a network drive and it is your home drive when you login to the system. This should be a convenient folder to save your files in these sessions although you can choose something else if you prefer.

One way to start the Matlab editor to create a new file with a given name is to type the following in the command window.

```
edit sess1a.m
```

If the file does not exist yet then this starts the editor for the creation of a file with the name given which will be saved in your current folder. This mode of doing things is likely to be quicker than clicking the mouse on `File` (top left hand corner), selecting `New` and then making further selections but you can choose what you do. The name of the file must be a valid name, i.e. it must start with a letter and with other letters, digits or the underscore character as was the case with naming variables. The name of the file should not be the same as an existing Matlab command or function although you are not prevented from doing this but you will get warnings advising that you use a different name. Please note that with the Windows operating system that we have in the labs the names of files is not case-sensitive and this is different to what is the case if you use a Mac or Linux. We advise that you only use lower case letters in the name of m-files.

Exercise 1.5.1

Type the following commands into your new script:

```
a=10
b=a+5
c=b-a
```

You save the file by clicking on the diskette sign and note that the existence of the file will be indicated in the left hand side panel. Keep the editor window open. There are several ways that you can run the file which you should try.

1. Type `sess1a` at the command prompt. This way of doing things has the advantage that you see the output from the script in the command window immediately below what you typed and also it does need that the file is open in the editor.
2. Press the `F5` key. This is probably the quickest option and it has the affect of ensuring that what is in the editor is saved and then the instructions are run with the output appearing in the command window. There can sometimes be a disadvantage in doing things this way as it is not always immediately clear which of the output shown is from the latest press of `F5`. When this is the case type `clc` in the command window first.
3. An alternative to the previous item is to just click the mouse on the green arrow symbol on the top bar.

Answer of exercise 1.5.1

The output in the command window should be as follows.

```
a =
    10
b =
    15
c =
     5
```

Exercise 1.5.2

Close the editor. Click the mouse on the name of the file used in the previous exercise as it appears in the left hand side panel, i.e. in the current folder panel. This should start the editor with that file. Modify so that it now contains the following statements.

```
a=10;
b=a+5;
c=b-a
disp('The value of c is')
disp(c)
```

Save the file with the name `sess1b.m` and run the file. The saving with a different name can be done by clicking on `File` and then selecting `Save As` and then entering the required name. The semi-colon in the first two statements means do the statements but do not display any output. `disp` is a for displaying what is between the round brackets and in the first case the display is just a string. Matlab uses the single quote character `'` for starting and ending strings.

Answer of exercise 1.5.2

The output is as follows.

```
c =
     5
The value of c is
     5
```

Adding comments

When your scripts become larger, you may want to start adding comments to remind yourself what specific parts of the script do. This is a highly recommended practice, because it is often difficult to remember few months later, what you were trying to achieve by these commands. You can add comments to a Matlab program by using character `%` — everything you type after this character on the same line will be ignored by Matlab. However, for readability, I would recommend that you generally avoid putting comments on the same line as statements but instead put them before the lines which they describe as is illustrated in the following example to end this first session.

Let us solve a general quadratic equation

$$ax^2 + bc + c = 0\,.$$

You should remember that two solutions of this equation can be found using the standard formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\,,$$

where the mathematics shorthand means

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}.$$

The following script implements this formula, for some pre-defined values of a, b and c:

Exercise 1.5.3

Use the editor to create the following script and execute it.

```
% coefficients of the quadratic
a = 2; b = -6; c = -8;

% let d be the discriminant and s its square root
d = b^2 - 4*a*c;
s = sqrt(d);

% now use the formula for the two values
x1 = (-b-s)/(2*a);
x2 = (-b+s)/(2*a);

disp('two solutions of this quadratic are')
disp(x1)
disp(x2)
```

Notice the comment lines and also notice that you can have blank lines which may help the readability. Another feature here is that the line for assigning a, b and c has 3 statements on the line. It is generally not recommended to have more than one statement per line (it usually makes harder to follow) but in situations such as the above the authors of these notes do not object.

As a final comment, if a, b and c are changed then the only restriction is that $a \neq 0$. The program still works when $a \neq 0$ and $d < 0$ in which case x1 and x2 are complex numbers. Complex numbers are covered in MA1710.

Answer of exercise 1.5.3

The output that this generates is as follows.

```
two solutions of this quadratic are
    -1
     4
```

Exercise 1.5.4

Modify the script from the previous exercise to solve the following quadratic equation:

$$x^2 - 7x + 6 = 0\,.$$

The correct answers should be 1 and 6.

Answer of exercise 1.5.4

All you need to do is to change the line which assigns values to a, b and c with the complete file being as follows.

```
% coefficients of the quadratic
a = 1; b = -7; c = 6;

% let d be the discriminant and s its square root
d = b^2 - 4*a*c;
s = sqrt(d);

% now use the formula for the two values
x1 = (-b-s)/(2*a);
x2 = (-b+s)/(2*a);

disp('two solutions of this quadratic are')
disp(x1)
disp(x2)
```

1.6 Summary

By the end of this session you should have mastered the following:

- Basic operation of Matlab, esp. use of Command Window and some of the other panels;
- Using the Command Window for computing mathematical expressions;
- Understanding the precedence of operators in Matlab;
- Knowing how to get help on Matlab commands;
- Using variables and being able to monitor their values via the Workspace panel;
- Creating Matlab scripts;
- Knowing how to output results from scripts into the Command Window;
- Being able to write scripts to implement a group of commands.

1.7 Further remarks

Our discussion of the floating point numbers, if you think about it carefully, implies that most of the results you are going to obtain in Matlab are not actually fully precise. This is not a mistake, this is the price that we pay for the convenience of doing numerics on the computer. The following exercise illustrates the issue:

Exercise 1.7.1

Compute the following expression:

```
1/3-(1-2/3)
```

It is easy to check, on a piece of paper, that the correct answer is 0. However, the answer returned by Matlab is `-5.5511e-017` (if you are using `short` numeric format) or `-5.551115123125783e-017` (if you are using `long` numeric format), i.e. a tiny non-zero number. There are many implications of this behaviour, which will be discussed in more detail when you will be learning about numerical methods.

1.8 Supplementary materials

MathWorks, the company that makes Matlab, provides an extensive on-line support for the system. In particular, they prepared a number of on-line tutorials for someone learning Matlab:

`http://uk.mathworks.com/help/matlab/index.html`

To understand Matlab in more depth, you may also consider getting one of the following books:

- Brian D. Hahn and D. T. D. T. Valentine. *Essential MATLAB for engineers and scientists*. Academic Press, 5th edition, 2013.
  `QA297.H345`. Only the 4th edition is currently in the Brunel library.

- Timothy A. Davis. *MATLAB primer*. CRC Press, eighth edition, 2011.
  `QA297.D38`.

The cost of purchasing standard Matlab license for your personal computer is high; however, the student version is priced more reasonably at £ 29. The student version of Matlab Suite sold for £ 55 is targeted at engineering students and not worth purchasing for a maths student.

There are also open-source alternatives to Matlab. These are programs that implement functionality similar to Matlab, although, naturally, with slight differences in both language and user interface. The differences are relatively minor; if your understanding of Matlab is good, you should not have substantial difficulties using one of these programs as an alternative to Matlab for your home computer. The web sites for the "Matlab clones" are as follows.

- Octave: `http://www.gnu.org/software/octave/`
- FreeMat: `http://freemat.sourceforge.net/`
- Scilab: `http://www.scilab.org/`

There is also some further comments about installing the first two from the following URL.

`http://people.brunel.ac.uk/~icstmkw/ma1710/index.html`