Computer Lab 1: Introduction to MATLAB

1.1 Introduction

Today you will be introduced to MATLAB: a powerful computational mathematics program used in science, engineering, commerce and the mathematical sciences all over the world.

While the main strength of MATLAB lies in numerical linear algebra, it can be used to plot functions of one variable, polar plots, surface plots and contour plots for functions of two variables. In addition, some symbolic manipulation is possible.

At several points in the lab, you will be asked to think, discuss, and guess what will happen before you type. Make sure you actually make a guess! It doesn't matter if you are wrong.

1.2 Getting Started with MATLAB

Use one of the options on the MAST10007 LMS website for how to access MATLAB.

Start MATLAB. When MATLAB opens, you should see several panes:

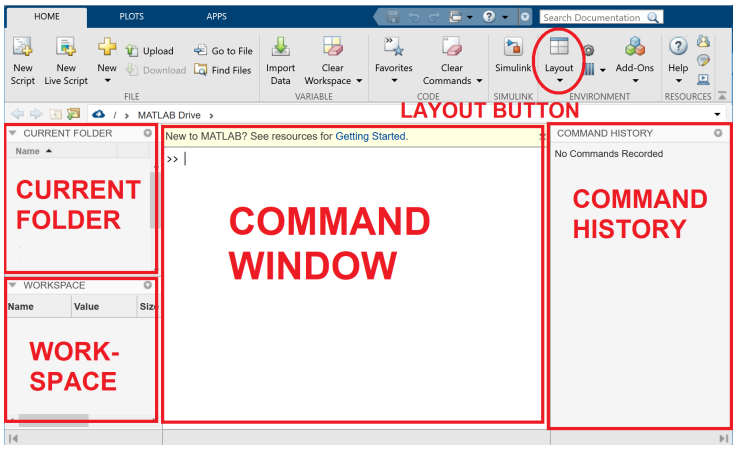
Command Window (center). The largest area is the *Command Window*. You can type commands next to the prompt » and the result will be displayed below the command.

Workspace. The *Workspace* window shows all the variables we have defined during the MATLAB session. You can double-click on a variable to edit its value.

Command History. The *Command History* area displays commands you have previously used. You can copy and paste commands from this window to the *Command Window*.

Current Folder. This pane shows any files that have been loaded in to MATLAB. We will use this pane in later lab classes.

If some of these panes are not visible in your window, you can show them using the "Layout" button from the MATLAB toolbar.



Note: the layout of the panes might vary depending on your MATLAB version.

To see an example of the sort of things MATLAB can do, type

```
»vibes
```

into the command window. Note that you should not type the » from the above line!

Remark: A full description of the features of MATLAB can be found in the online help. To access this, click the '**?**' icon in the MATLAB toolbar and then click the 'MATLAB' link.

1.3 Exercises

We will use the image compression example from Q4 in Tutorial 1 as an excuse to explore the manipulation of matrices in MATLAB.

Exercise 1: Entering Data

Let's start at the basics: we will enter a variable x, and give it the value 3. Type
`»x=3`

You have entered your first variable. Notice that it appears in the Workspace window. Now whenever you type `x`, MATLAB will recognise it is the number 3. Let's enter the following matrix:

$$M = \begin{bmatrix} 1 & 3 \\ 8 & -7 \end{bmatrix}$$

To enter this matrix, we need to type in the information in a way MATLAB understands. So, we need to learn the *syntax* for entering a matrix.

In the command window, type
`»M = [1 x; 8 -7]`

Note that:

- Square brackets start and end the matrix;
- Spaces separate entries;
- A semicolon starts a new row.

We could have easily typed 3 instead of `x`, but this shows how MATLAB now recognises `x` as being 3. Notice that the variable M has now appeared in the Workspace window.

Try to enter the following matrix from Tutorial 1 into MATLAB:

$$A = \begin{bmatrix} 5 & 10 & 15 & 10 & 5 \\ 3 & 10 & 13 & 10 & 3 \\ 13 & 1 & 10 & 1 & 13 \\ 15 & 10 & 0 & 10 & 15 \end{bmatrix}$$

When we need to use vectors in MATLAB they are entered as column matrices, for instance:

$$\mathbf{v} = \begin{bmatrix} 1 \\ -6 \\ 9.3 \\ 4 \end{bmatrix}$$

is entered as the column matrix `v = [1; -6; 9.3; 4]`.

Enter the first three standard basis vectors e_1, e_2, e_3 of $\mathbb{R}^4$ into MATLAB variables `e1, e2, e3`.

Finally, MATLAB is designed to deal with *complex numbers* with ease. To enter complex numbers, the syntax is pretty easy.

Type
`»z = 3+5i`
to enter the complex number $3 + 5i$.

Exercise 2: Arithmetic Operations

To simplify life, get MATLAB to compute the matrices U, S, V featuring in Q4, Tutorial 1, by typing
`[U, S, V] = svd(A)`

MATLAB uses a wide range of operators. We will introduce a few of them here.

The multiplication operator *. This will automatically do normal, scalar, or matrix multiplication depending on what variables you are trying to multiply.

Use this operator to find (i) $3A$ (ii) US (iii) SU

$$3A = \begin{bmatrix} 15 & 30 & 45 & 30 & 15 \\ 9 & 30 & 39 & 30 & 9 \\ 39 & 3 & 30 & 3 & 39 \\ 45 & 30 & 0 & 30 & 45 \end{bmatrix} \qquad US = \begin{bmatrix} -19.8375 & -9.0124 & -0.4986 & 0 & 0 \\ -17.0005 & -9.8098 & 1.3223 & 0 & 0 \\ -17.8623 & 6.5199 & -8.8559 & 0 & 0 \\ -22.4438 & 10.2075 & 6.4872 & 0 & 0 \end{bmatrix}$$

SU does not exist; we cannot multiply a 4×5 matrix by a 4×4 matrix.

The addition operator +. This operator will do normal scalar addition, or matrix addition depending what variables you are trying to add.

Two matrices of the same size can be added together by simply adding the corresponding entries.

First do the calculation by hand:

$$A + S = \begin{bmatrix} 43.7986 & 10.0000 & 15.0000 & 10.0000 & 5.0000 \\ 3.0000 & 28.0044 & 13.0000 & 10.0000 & 3.0000 \\ 13.0000 & 1.0000 & 21.0684 & 1.0000 & 13.0000 \\ 15.0000 & 10.0000 & 0.0000 & 10.0000 & 15.0000 \end{bmatrix}$$

Now add them on MATLAB by typing
```
»A+S
```

The transpose operator '.

Type `A'`. What did it do?

$$A' = A^T = \begin{bmatrix} 5 & 3 & 13 & 15 \\ 10 & 10 & 1 & 10 \\ 15 & 13 & 10 & 0 \\ 10 & 10 & 1 & 10 \\ 5 & 3 & 13 & 15 \end{bmatrix}$$

Use MATLAB to compute (i) UU^T; (ii) VV^T; (iii) USV^T. What do you notice? UU^T is the 4×4 identity matrix, and VV^T is the 5×5 identity matrix. USV^T gives A as expected.

Exercise 3: Be careful what you wish for

Is it possible to add the matrix A to the scalar 2?
No, a matrix can only be added to another matrix of the same dimensions.

What do you think MATLAB will do if you try to add them?
Give an error message?

Try it. Did it give an error? What did MATLAB actually do?
No error message, instead it added 2 to every entry of A.

The lesson is to always know how an operator works, and not to assume it will do exactly the same thing that we would expect mathematically.

Exercise 4: More matrix multiplication

Compute Ue_1, Ue_2, Ue_3. What do you notice?
Ue_i is a vector corresponding to the ith column of U.

Building on the observation made above, define the matrix `E = [e1 e2 e3]` and let `Unew = U * E`. Can you describe in words the relation between `Unew` and `U`?
U_{new} is the 4×3 matrix given by the first three columns of U.

Type `Snew = diag([38.7986, 18.0044, 11.0684])`. What do you think the command `diag` does?
It creates a diagonal matrix with diagonal entries $38.7986, 18.0044, 11.0684$, and all other entries 0.

How are the matrices `Snew` and `S` related?
S_{new} is the matrix obtained by deleting the last row and last two columns of S.
These columns contained all zeroes.

Exercise 5: Extracting Entries from Matrices

We can **find the value of a specific entry** in the matrix using row-column notation.

Type
```
»A(1,2)
```
Output is 10

Notice that it tells us the entry in the first row, and second column. This can be very useful when we are dealing with incredibly large matrices. Also notice that since we didn't assign a variable to the answer, MATLAB automatically assigns it to '`ans`'. MATLAB will overwrite `ans` next time, so always assign to a variable if you want to use it later.

We can also **extract a whole column**.

Type `v=A(:,2)`

Output is $\begin{bmatrix} 10 \\ 10 \\ 1 \\ 10 \end{bmatrix}$

Important features are:

- the colon picks all rows
- the 2 picks the 2nd column

What would you need to type to extract the 3rd row? Try it out.

```
v=A(3,:)
```

A useful operator is the **colon operator**.

Type in the following commands and figure out what it does.
(i) `2:8` (ii) `3:0.5:6`
(i) Output is $\begin{bmatrix}2 & 3 & 4 & 5 & 6 & 7 & 8\end{bmatrix}$ (ii) Output is $\begin{bmatrix}3 & 3.5 & 4 & 4.5 & 5 & 5.5 & 6\end{bmatrix}$

Investigate further by trying your own combinations. When you think you've figured it out, use the colon operator to create the row matrix $\begin{bmatrix}1 & 1.2 & 1.4 & 1.6 & 1.8 & 2.0 & 2.2 & 2.4\end{bmatrix}$
What command did you use?
`v=1:0.2:2.4` or `v=[1:0.2:2.4]`

Combining a couple of the concepts we tried above, we have a way of extracting a matrix sitting inside another matrix.

For instance, check that `Snew = S(1:3, 1:3)` gives the same matrix `Snew` that we calculated above.

Use this technique to compute (i) the 3×5 matrix `R` obtained from `S` by deleting the last row; (ii) the matrix `Vnew` that consists of the first three columns of `V`.
`R = S(1:3, 1:5)` or `R = S(1:3, :)`, `Vnew = V(1:5, 1:3)` or `Vnew = V(:, 1:3)`.

You now have all the ingredients to use MATLAB to answer part (a) of Q4 in Tutorial 1:

Verify that `U * S` is the same matrix as `Unew * R`.

Verify that `R * V'` is the same matrix as `Snew * Vnew'`.

Verify that `A` is the same matrix as `Unew * Snew * Vnew'`.

Pause and admire the beauty of this conclusion: even though `Unew`, `Snew`, `Vnew` are smaller than the original matrices `U`, `S`, `V`, the result is the same matrix `A`.

Optional challenge: Implement the rest of Q4 from Tutorial 1 in MATLAB and see what happens.

Exercise 6: Correcting Data

Input the matrix $B = \begin{bmatrix}-8 & 1\\ 1 & 9\\ 0 & 3\end{bmatrix}$

Suppose that you made a mistake while typing, and you meant to enter

$$B = \begin{bmatrix}-8 & 73\\ 1 & 9\\ 0 & 3\end{bmatrix}.$$

You can correct the error in several ways:

(1) **Using the command line.**

You can retrieve previous commands using the up arrow key.

(i) Press ↑. You will see your previous command appear.

(ii) Change the 1 to 73.

(iii) Press Enter.

(2) **Using the *Command History* window.**

(i) Check that the *Command History* pane is visible on the bottom-right of the MATLAB window. This window shows all the commands you have entered.
If the Command History pane isn't showing, then select the HOME tab in the toolbar at the top of the MATLAB window, click the Layout button, then 'Command History' → 'Docked'.

(ii) Select the command `B=[ -8 1; 1 9; 0 3 ]` in the *Command History* window.

(iii) Copy and paste the command into the *Command Window*.

(iv) Change the entry in B and press Enter.

(3) **Changing a single entry in B.**

Suppose that we meant to enter $B = \begin{bmatrix} -8 & 13 \\ 1 & 9 \\ 0 & 3 \end{bmatrix}$.

We need to change the entry in row 1, column 2.

Enter the command
```
»B(1,2)=13
```

The $B(1,2)$ refers to the 1st row and 2nd column of B. This command changes just the entry in row 1, column 2 and leaves the rest of B unchanged.

(4) **Using the *Workspace* window.**

Look in the *Workspace* window to see the matrices and variables we have entered so far. Double-click on the entry for B. This will open a spreadsheet window showing the entries in B. You can edit the entries of B in this window.

Change B to $\begin{bmatrix} -8 & 5 \\ 10 & 9 \\ 0 & 53 \end{bmatrix}$

When done, you can close the spreadsheet window by clicking the 'X' in the top-right corner.

Exercise 7: Data Entry Shortcuts

Just like in mathematics, in MATLAB a **function** takes one or several **arguments**, and outputs a **result**. The arguments can be all sorts of things: scalars, matrices, even files. Similarly, the outputs can be things like graphs, files, scalars and more.

Functions in MATLAB all use a common syntax, very similar to the way we use function in mathematics. The name of the function goes first, followed immediately by round brackets. Inside the brackets go the arguments, separated by commas. The arguments must go in the right order.

Let's begin with some basic functions which create new matrices.

Type `eye(4)`. What does this function do?
Creates a 4×4 identity matrix.

Type `ones(6,3)`. What does this function do? What do the two arguments represent?
Creates a 6×3 matrix with all entries 1.

Type `ones(3)`. What does this function do?
Creates a 3×3 matrix with all entries 1.

Type `zeros(4,1)`. What does this function do?
Creates a 4×1 matrix with all entries 0.

MATLAB can also *concatenate* matrices if they have compatible sizes.

Type `[zeros(3,2) ones(3,5)]` to see what happens.
It concatenates the matrices `zeros(3,2)` and `ones(3,5)` horizontally:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

What about `[zeros(2,3); ones(5,3)]`?
It concatenates the matrices `zeros(2,3)` and `ones(5,3)` vertically.

What happens if you try to concatenate two matrices with incompatible sizes?

Try the command `[zeros(2,2); ones(3,3)]` to see.
MATLAB gives an error message:
'Dimensions of arrays being concatenated are not consistent.'

Exercise 8: MATLAB m-files

In this subject we will only use a small portion of the features of MATLAB. In this section you can try out a MATLAB game. While the graphics may seem simple, they are the product of quite sophisticated matrix manipulation.

Instead of writing all your commands in the MATLAB window, it is possible to prepare them in a file called an *m-file* (so called because each file has the extension *.m*). You will learn how to write these files later in the semester.

The program *xpbombs* is a game stored in the form of an m-file. Try entering

```
»help xpbombs
```

The result is a description of how to play the game.

Now enter

```
»xpbombs
```

Try clicking on one of the squares to find out what happens.

The aim of the game is to work out which squares contain bombs and put a flag on them, without detonating any of the bombs.