Introduction to $\LaTeX$and summary of basic functions

Dr Gavin M Abernethy

1 What is LaTeX?

LaTeX is a typesetting language specifically designed for creating mathematical documents. Documents prepared in LaTeX must be compiled, meaning that you write the source code (in a file with the .tex extension) and then run a program that interprets the .tex file to create the PDF output.

2 Set up

LaTeX can be written using any plain text editor (Wordpad, Notepad, Sublime, Visual Studio, etc.). However, you will need to install a dedicated program that can interpret this data and compile the PDF.

3 Programs for compiling .tex files

There are numerous free software options for this. I recommend choosing one of two methods:

- Local installation:
  1. First install a "TeX distribution" - either TeX Live (`https://www.tug.org/texlive/`) or MikTeX (`https://miktex.org/howto/install-miktex`).
  2. *Then*, if it was not already included in the previous package, install an editor such as TexMaker (`https://www.xm1math.net/texmaker/`).
- Alternatively (and much more easily!), you can create an account on Overleaf (`https://www.overleaf.com/`). This is a cloud-hosted browser-based solution that many staff use and which does not require any installation. Simply register an account, then go to "New Project", "Upload Project", and choose the compressed folder `minimal_template.zip`

There are also numerous other free software choices. Those popular amongst the mathematics staff include:

- TeXworks (`http://www.tug.org/texworks/`)
- Sublime text editor in combination with a plug-in called latextools (`https://www.sublimetext.com/`)
- TeXnicCenter (`https://www.texniccenter.org/`)
- TeXStudio (`https://www.texstudio.org/`)

Most of these options are available free for both Windows and MacOS, and (like TeXMaker) they will also require installing a "TeX distribution" such as MikTeX (`https://miktex.org/howto/install-miktex`) *beforehand*!

3.1 Document class

The first decision to make in a .tex file is the document class - this is essentially which template the document should load. Common options include the `article` class (this is used for anything simple such as notes, tutorial solutions, preprint research articles, etc.), the `thesis` and `book` classes, and `beamer` which is used for creating PowerPoint-style slides. Many research journals produce their own article classes, so that submissions can be formatted into their desired form.

For the project report, we will stick with the default `article` class.

3.2 Packages

After selecting the class, the next few lines of a .tex file will state which extra packages should be loaded. For example:

```
\usepackage{pdfpages}
```

Loading this package allows you to use a command that incorporates other PDF documents into you own document.

Similar to importing packages in Python, packages contain extra commands that you can access. In the provided template, we have loaded the most common packages (e.g. `amsmath` and `amssymb` which enable maths symbols). Advanced users may wish to load other packages to access more options, although note that conflict between packages (e.g. that both try to define the same command) is a common source of errors. Particularly useful packages include `hyperref` and `enumitem`, although there are many others.

For final year projects, we have created a custom package and loaded it in the template. This sets out the frontmatter, appendices, and the two-column format for the main body of the report.

3.3 Frontmatter

Following this, in most document types, you can state the title, the author and the date. Then begin the document and generate the title page. This is achieved by (at minimum):

```
\title{Introduction to LaTeX and summary of basic functions}
\author{My name}
\date{\today}
\begin{document}
\maketitle
```

The end of the document will need to contain the closing line:

```
\end{document}
```

4 Typing in LaTeX

Like any programming language, you can comment out lines. This is done using the percentage symbol:

```
 % this is a comment
```

Commands in LaTeX are denoted by the backslash, and the argument is passed in curly brackets, as you can see from the examples above.

Typing in LaTeX is separated into two main modes - you are always in either *text mode* or *math mode*. Both appear differently, and have their own commands that are only recognised when in the appropriate mode.

4.1 Structure

You should break your document up into numbered sections. To create a section with the title "Introduction", we can use `\section{Introduction}`, and to add a subsection (i.e. numbered 1.1, 1.2, 1.3, etc.) use `\subsection{Name of this subsection}`

In all cases, LaTeX will figure out the appropriate numbering - so I did not have to name this current subsection 4.1, that was determined for me, and if I inserted or removed some (sub)sections it would automatically update the numbering to be correct.

Sometimes we would prefer to have a section title, but without the numbering. This can be achieved by including an asterisk after the command. For example, we usually don't number the references section:

`\section*{References}`.

4.2 Text mode

You will be in text mode by default. In this mode, you can type regular text like any word processing software. Note that LaTeX will decide how to space out words, so including say ten spaces between two words will actually have no effect.

Commands for this mode include formatting the text:

- I would like to `\emph{italicize this}` = I would like to *italicize this*
- I would like to `\textbf{bold font this}` = I would like to **bold font this**

4.2.1 Manipulating space

We can force a new page using `\newpage`, or take a new line using two backslashes `\\`. If you get the error "no line here to end", such as from trying to introduce a linebreak immediately after a figure, or if you are trying to add two subsequent linebreaks, you can get around this by prefacing the backslashes with `~`. So to break two lines (such as the end of this paragraph!): `\\~\\`

Alternatively, you can create a "medium-sized" or "large" linebreak using the commands `\medskip` and `\bigskip`.

We can also manually attempt to create horizontal or vertical whitespace (for example to spread out parts of an equation) using `\hspace{5mm}` and `\vspace{5mm}`, although this can be a bit messy.

4.3 Lists

To add lists in the form of bullet points, we can begin the `itemize` environment. This, like anything that has a `\begin{}` must also be complemented by an `\end{}`. So for example:

```
\begin{itemize}
     \item The first bullet point here.
     \item The second bullet point here.
\end{itemize}
```

This creates the following output:

- The first bullet point here.
- The second bullet point here.

If you want to use numbered points, replace `itemize` with `enumerate`.

4.4 Math mode

When typing mathematics, we need to enter math mode. This is a bit like entering the equation editor in MS Word, but it is much more powerful and it is very easy to create composite commands to produce whatever we need.

Math mode can either be "inline", meaning that it occurs within a block of text. An example would be this equation $y = mx + c$ which, as you can see, is within this text. On the other hand, there are various ways of creating a standalone math environment, such as the equation:

$$y = ax^2 + bx + c$$

which was given its own line etc.

In math mode, spaces are completely ignored, so include them as you wish simply to increase the readability of the maths parts of your .tex file.

4.4.1 Inline mathematics

To insert inline maths, simply encase it in a pair of dollar signs.

For example:

"This is a line containing the equation $y = mx + c$"

is created by typing:

```
"This is a line containing the equation $y=mx+c$"
```

Inline math should only be used for very small equations or for discussing a variable such as displacement, x, that needs to be italicised and in "curly math font". Anything important, large, or requiring numbering should not be written inline. It also usually doesn't look very nice if the expression is too large, such as $\frac{1}{13-y}$.

4.4.2 Single equation

To type a single line of mathematics, centered in its own line, use the `equation` environment:

```
\begin{equation}
      y_{1} = a x^{2} + bx + c
\end{equation}
```

This creates a numbered equation:

$$y_1 = ax^2 + bx + c \tag{1}$$

Alternatively, we can include an asterisk after the command to avoid numbering:

```
\begin{equation*}
      y_{2} = a x^{2} + bx + c
\end{equation*}
```

This creates a non-numbered equation:

$$y_2 = ax^2 + bx + c$$

We can also enter a standalone line of mathematics using a backslash and square brackets:

```
\[
      y_{3} = a_{0} + a_{1}x + a_{2}x^{2} + \dots
\]
```

Which results in:

$$y_3 = a_0 + a_1x + a_2x^2 + \dots$$

4.4.3 Multi-line equations

To include mathematics consisting of multiple lines, instead of the `equation` environment we can use some alternatives such as `eqnarray` which allows us to align the different lines at one point (such as around an equality), or the `align` environment which can allow for multiple parts of the different lines to be in alignment. In any case, we use `\\` to break to the next line, and all lines will be individually numbered unless we include the asterisk (as before).

For an `eqnarray` example, where we align the equality symbol on each line:

```
\begin{eqnarray}
     x &=& 1 + 3 \times 2 \\
        &=& 1 + 6 \\
        &=& 7
\end{eqnarray}
```

Which results in:

$$\begin{aligned} x &= 1+3\times 2 && (2)\\ &= 1+6 && (3)\\ &= 7 && (4) \end{aligned}$$

Then for an `align` example, where we align *both* the x's and the y's on each line into two columns:

```
\begin{align}
     &x=5 &y=17 \\
     &2x =1 &-2y = 0
\end{align}
```

Which results in:

$$\begin{aligned} &x=5 & y&=17 && (5)\\ &2x=1 & -2y&=0 && (6) \end{aligned}$$

Finally, we could also use the `multline` environment to split up a single very long equation over two lines. In this case, the first line is left-aligned while the second is right-aligned.

For example:

```
\begin{multline*}
    y = a_{0}+a_{1}x+a_{2}x^{2}+a_{3}x^{3}+a_{4}x^{4}+a_{5}x^{5}+a_{6}x^{6} \\
        +a_{7}x^{7}+a_{8}x^{8}+a_{9}x^{9}+a_{10}x^{10}+a_{11}x^{11}+a_{12}x^{12}
\end{multline*}
```

Which results in:

$$y = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 \\ + a_7x^7 + a_8x^8 + a_9x^9 + a_{10}x^{10} + a_{11}x^{11} + a_{12}x^{12}$$

4.4.4 Common math commands

You have already seen some commands that can be used in math mode in the above examples:

- Indices, subscripts and superscripts: `$x_{1}$` $= x_1$; `$x^{2}$` $= x^2$
- Greek letters: `$\alpha, \beta, \gamma, \sigma$` $= \alpha, \beta, \gamma, \sigma$
- Fractions: `$\frac{x}{y}$` $= \frac{x}{y}$
- Common trigonometric functions: `$\sin(x), \cos(13)$` $= \sin(x), \cos(13)$
- Multiplication: `$ 13 \times a $` $= 13 \times a$
- Division: `$ 13 \div a $` $= 13 \div a$
- Derivatives: to obtain the upright font used for derivatives, wrap the "d" in `\mathrm{}`, so for example:

  `$\frac{\mathrm{d}y}{\mathrm{d}x}$` $= \frac{\mathrm{d}y}{\mathrm{d}x}$
- Integrals: `$\int \cos(x) \mathrm{d}x$` $= \int \cos(x)\mathrm{d}x$

An example that brings much of this together is the general homogeneous second order ODE:

$$a\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + b\frac{\mathrm{d}y}{\mathrm{d}x} + cy = 0$$

which we can create using:

```
\begin{equation*}
   a\frac{\mathrm{d}^{2}y}{\mathrm{d}x^{2}} + b\frac{\mathrm{d}y}{\mathrm{d}x} + cy=0
\end{equation*}
```

5 Other environments

Creating other objects, such as figures or tables, will require entering a new environment.

5.1 Figures

To insert a figure, you will want to make sure it is in the same folder as your .tex files, then we can use the `figure` environment.

In this example, we enter the figure environment, using the optional argument `[H]` to tell LaTeX to place the figure exactly *here* in the document. The `\centering` command then ensures that the figure is centrally-justified, and then we request the figure - stating that it should be scaled so that it is 80% of the linewidth. Finally, we add a caption, and close the figure environment.

```
\begin{figure}[H]
 \centering
 \includegraphics[width=0.8\linewidth]{lya1.png}
 \caption{Lyapunov exponents in a particular two-dimensional map}
\end{figure}
```

Resulting in:

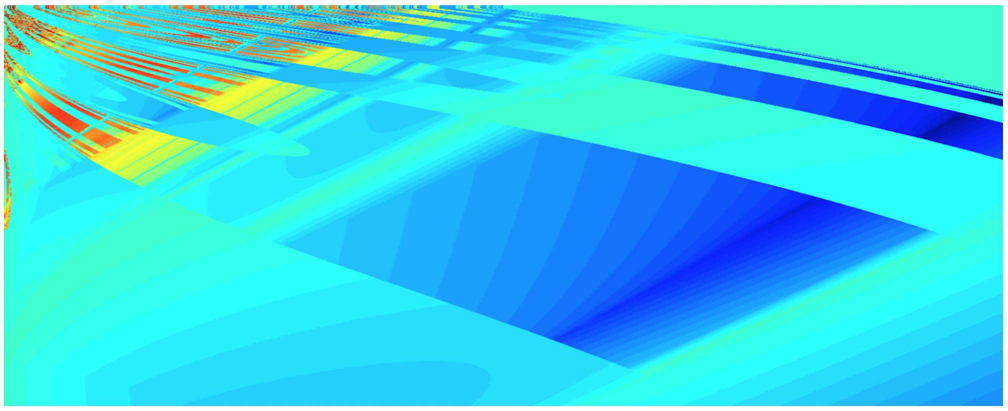

Figure 1: Lyapunov exponents in a particular two-dimensional map

5.2 Tables

When creating tables, we have a great deal of flexibility and control, however this comes at the cost of being rather more complex than in other programs such as word.

We can create a `table` environment, to control the location of the table, and then within that enter the `tabular` environment. When doing so, it is necessary to indicate how many columns there should be, and if a vertical line should visibly separate them. For example, to have five columns but only the first one separated by a vertical line, and with vertical lines around the outer edges of the table, you would write `\begin{tabular}{|c|c c c c|}`. Then you go line-by-line from top to bottom, separating the elements in each column (on the same line) by ampersands, and indicating where you would like any horizontal lines with `\hline`.

Consider this example:

```
\begin{table}[H]
 \centering
 \begin{tabular}{|c|c|}
     \hline
     \textbf{Region} & \textbf{Area ($m^{2}$)}\\
     \hline
     London & 1580000000\\
     South East & 19100000000\\
     South West & 23950000000\\
     East Midlands & 15600000000\\
     West Midlands & 13000000000\\
     East of England & 19100000000\\
     North East & 8600000000 \\
     Yorkshire and the Humber & 15400000000 \\
     North West & 14200000000\\
     Wales &  20800000000 \\
     Scotland &  78800000000 \\
     \hline
 \end{tabular}
 \caption{Geographic area of the regions of Great Britain}
\end{table}
```

Which results in:

Region	**Area** (m^2)
London	1580000000
South East	19100000000
South West	23950000000
East Midlands	15600000000
West Midlands	13000000000
East of England	19100000000
North East	8600000000
Yorkshire and the Humber	15400000000
North West	14200000000
Wales	20800000000
Scotland	78800000000

Table 1: Geographic area of the regions of Great Britain

6 Debugging

Very often, your PDF will fail to compile and the program you are using will present you with a set of angry red text.

LaTeX, like any programming language, does not tolerate any errors and will fail if there is anything missing or incomplete. General principles of solving bugs that may arise include:

- Have you ended every environment that you began? i.e. For every `\begin{equation}` do you have an `\end{equation}` to match it? Similarly, dollar signs should occur in pairs, and curly brackets always need to be closed.
- Have you spelled commands correctly?
- Have you tried to use a command in the text environment that only works in the math environment?

7 Referencing

To reference in LaTeX, you will need to have a separate file with the extension `.bib` containing the details of each reference in a very specific format, using resembling this example:

```
@article{may1976simple,
author = {May, Robert M},
date-added = {2022-07-07 15:16:20 +0100},
date-modified = {2022-07-07 15:16:20 +0100},
journal = {Nature},
pages = {459},
title = {Simple mathematical models with very complicated dynamics},
volume = {261},
year = {1976}}
}
```

Fortunately you can usually download a reference for an individual article in this format directly from the article's listing on the journal website, or from the Google Scholar hit for the paper. Look for "Cite this paper" or something similar, and choose the "BibTeX" option. This will download a file containing the reference in the above form, and you can add it to your `.bib` file containing all of your references.

You may also wish to use specialist software, such as Bibdesk, Refworks or Mendeley to help you with this if you are thinking of going on to a MSc or PhD where you will have hundreds or thousands of references to manage. These tools store all of your references in a searchable database.

Now that you have a copy of the reference, how do you actually insert it into your paper?

As with everything else, ensure that the `.bib` file containing the reference (which we shall call `references.bib`) is in the same folder as your `.tex` file. Then look at the first line of that paper's reference. The name `may1976simple` is the unique identifier, which we use to insert the reference using the `\cite{}` command.

Thus:

"`This \cite{may1976simple} is an interesting paper.`"

results in:

"This [May76] is an interesting paper."

However, this won't work until we include the following commands at the end of our document:

```
\bibliographystyle{apacite}
\bibliography{references}
```

The first of these tells LaTeX that we want the reference in the final bibliography to be in the APA style (this is a requirement), and the second line indicates when we want to create the Bibliography featuring all of the references that we have used in the document. The argument `{references}` tells LaTeX that it can locate the details of the reference in our file `references.bib`

Finally, note that we many need to run our compiler program up to 4 times to get the references working correctly:

1. Once using the `pdflatex` or `Quick Build` option (varies by program).
2. Once using the `BibTeX` option, as it looks for the correct `.bib` file.
3. Again using the `pdflatex` or `Quick Build` option (varies by program).
4. Possibly again using the `pdflatex` or `Quick Build` option (varies by program)!

The results of these commands should produce the in-text citation above, and the Bibliography below:

References

[May76] Robert M May. Simple mathematical models with very complicated dynamics. *Nature*, 261:459, 1976.