Design and Computing 1

Introduction to Scientific Computing

Daniel Poole
Department of Aerospace Engineering
University of Bristol
`d.j.poole@bristol.ac.uk`

2017

LECTURE 10

Intro to Scientific Computing

- Have learnt the basics of programming including variables, loops, conditionals, files and arrays.
- Have learnt programming in C and MATLAB
- Have learnt to create figures using Tecplot

TODAY

- Processing results and documenting
- Introduction to LaTeX
  - Open-source scientific document processing software
  - Basics of LaTeXincluding writing technical documents

Technical Documenting

During the course we have learnt how to program a computer to find a solution to a given problem that would otherwise have been to difficult or time-consuming to do by hand. We have also learnt how to use the basics of a very powerful piece of scientific plotting software called Tecplot.

The final aspect of the course is to introduce how we document our findings into a report which we can deliver, whether this be to our manager, supervisor, publisher, etc. Let's go back to the 'engineering method':

$$\text{PROBLEM} \Rightarrow \text{SOLVE} \Rightarrow \text{REPORT}$$

In this lecture we will introduce a language, called LaTeX, which can be used to produce well formatted and well structured technical documents.

LaTeX vs. Word

Up until this point, most of you will have produced documents using Microsoft Word. However, while Word can be useful in certain circumstances, it is best described as a **jack of all trades, master of none** piece of software.

When trying to write a technical document in particular, issues with Word include, though are not limited to:

- Difficult to structure and format effectively
- Poor at writing equations
- Major difficulty in bibliography, cross-referencing and figure labelling
- Lack of compatibility between versions !!
- Cost

LaTeXvs. Word

The purpose of this lecture is to introduce you to a **FREE** alternative to Microsoft Word, called LaTeX.

LaTeXis a mark-up language used to create scientific documents. Similar to writing and compiling a C program, a LaTeXdocument is simply a text file with a series of commands that has to be 'compiled' to produce an output document. You can use any standard text editor to create a LaTeXdocument, however there are dedicated softwares out there that can help e.g. Texmaker.

The difficulty in LaTeXis the learning curve. We will only scratch the surface of how to create a document. There are a number of examples of blackboard of scientific documents that have been created using LaTeXwhich will give more detail in some of the more minor syntax found in the language

LaTeX Program Anatomy

```
\documentclass{article}
```

The type of document

```
\usepackage{amsmath}
\usepackage{graphicx}
```

Load some packages to use

```
\begin{document}
```

Start main body of document

```
We can write a document as normal
here.  When we compile this document
  ,
the LaTeX compiler will create a
number of outputs including the
final pdf document.

We can also write equations.

\begin{equation}
a=b+c
\end{equation}
```

Body

```
\end{document}
```

End main body of document

LaTeXProgram Anatomy

`\documentclass{article}`

LaTeXhas different basic types of document. The most common are `article`, but we can also have classes such as `letter`, `book`, `report`. The `article` class is almost always used. Sometimes, user written class functions can also be used, particularly for journal papers.

`\usepackage{}`

To use various functionality, we need to load pakages (like loading libraries in C). Common packages are for loading images, entering mathematical formulae and formatting tables.

`\begin{document}` and `\end{document}`

Start and end the main body of the document. Generally, all text should be written inside these comments.

Writing Text

To write text, separate words with spaces, separate sentences using periods, and separate paragraphs by blank lines.

```
We write a latex document as we would in any normal document.  Of course
   the
main difference is that the compiler takes care of te formatting for us.
This saves us epochs of time by avoiding having to perform minutia changes
to the formatting of our documents that word processors such as Microsoft
Word require.

To start a new paragraph, we must ensure we leave a blank line between the
previous and the new paragraph.  This tells the compiler to start a new
paragraph.  The compiler will either indent or create a line break
   depending
on the class of document that has been selected in the first line.

\noindent If we start a new paragraph and don't want to indent it (maybe
after we have written an equation), then we can use the noindent command,
as shown here.
```

When compiling, if multiple spaces are together, the compiler accepts these as just one continuous space.

Text Formatting

To bold and italic pieces of text we can:

```
We want to \textbf{bold some text}, and \textit{italic some text}.
```

The default font size in the `article` class is 10pt. The font size can also be changed from the default using:

`\normalsize{}`	Normal size	`\normalsize{}`	Normal size
`\small{}`	Small	`\large{}`	large
`\footnotesize{}`	Foot note size	`\Large{}`	Large
`\scriptsize{}`	Script size	`\LARGE{}`	LARGE
`\tiny{}`	Tiny	`\huge{}`	huge
		`\Huge{}`	Huge

Sectioning and Structure

Scientific reports are usually structured in a logical numbering fashion, where there are sections and sub-sections within those sections::

1. First Section
1.1. Sub-section
1.1.1 Sub-sub-section
1.1.2 Sub-sub-section
1.2. Next sub-section
2. Next section

1. C Programming
1.1. Basics of Programming
1.1.1. Compiling
1.1.2. Variable Types
1.2. Loops
2. MATLAB Programming

The numbering is automatically handled by LaTeX, we just need to use `\section{}`, `\subsection{}` and `\subsubsection{}`:

```
\section{C Programming}
\subsection{Basics of Programming}
\subsubsection{Compiling}
\subsubsection{Variable Types}
\subsection{Loops}
\section{MATLAB Programming}
```

```
1    C Programming

1.1    Basics of Programming

1.1.1    Compiling

1.1.2    Variable Types

1.2    Loops

2    MATLAB Programming
```

Lists

Bullet points lists and numbered lists are simple in LaTeX:

```
\begin{itemize}
\item First element of list
\item Second element of list
\begin{itemize}
\item Nested list
\item Next item in nested list
\end{itemize}
\item Final element of list
\end{itemize}
```

- First element of list
- Second element of list
  - Nested list
  - Next item in nested list
- Final element of list

```
\begin{enumerate}
\item First element of list
\item Second element of list
\begin{enumerate}
\item Nested list
\item Next item in nested list
\end{enumerate}
\item Final element of list
\end{enumerate}
```

1. First element of list
2. Second element of list
   (a) Nested list
   (b) Next item in nested list
3. Final element of list

Tables

To create tables in LaTeX, it is recommended to load the `booktabs` package. This will create nicer looking tables that are better spaced.

```
\usepackage{booktabs}
```

To create a table, we use the `tabular` environment. The specification of the columns (i.e. justification) is then done using entries in a {} entry (see below example). Use & to separate entries in a row, and \\ to create a new line in the table.

```
\begin{tabular}{l c c}
\toprule
Variable & Value & Units \\
\midrule
x        & 10    & m     \\
F        & 25    & N     \\
\bottomrule
\end{tabular}
```

Variable	Value	Units
x	10	m
F	25	N

Best table aesthetics are obtained by not having vertical lines, and just having horizontal lines.

Figures

To load figures, we must first create the figure, using say Tecplot. When choosing the output format, vector graphics (pdf or eps) will produce higher quality results, but care must be taken to import these. Pixel-based images are simple to import, but suitable quality of the figure is important.

To load figures, we need to load `graphicx`:

```
\usepackage{graphicx}
```

To include a figure called `myfig.jpg` that has width half the total width of the page, we use:

```
\includegraphics[width=0.5\linewidth]{myfig.jpg}
```

Floats: Captions and Placement

We can encapsulate figures and tables within a floating environment. This means LaTeXcan decide where the best place to place them within the document is, and this is the most common way of including figures and tables. This also allows us to add a reference to the figure, and also captions.

Figures

Everything is wrapped within the `figure` environment.

```
\begin{figure}[h!]
  \includegraphics[width=0.5\linewidth]{myfig.jpg}
  \caption{This is a figure}
  \label{f:figurelabel}
\end{figure}
```

Using labels allows us to reference a specific figure in the text. LaTeXwill then take care of all of the numbering and ordering and cross-referencing of the figures and tables.

```
This is how to reference figure \ref{f:figurelabel} in the text.
```

Floats: Captions and Placement

We can encapsulate figures and tables within a floating environment. This means LaTeXcan decide where the best place to place them within the document is, and this is the most common way of including figures and tables. This also allows us to add a reference to the figure, and also captions.

Tables

The `tabular` environment is wrapped within the `table` environment.

```
\begin{table}[h!]
  \caption{This is a table}
  \label{t:tablelabel}
  \begin{tabular}{l c c}
    \toprule
    Variable & Value & Units \\
    \midrule
    x        & 10    & m     \\
    F        & 25    & N     \\
    \bottomrule
  \end{tabular}
\end{table}
```

Floats: Sub-figures

In many circumstances, we want a figure to have multiple sub-figures. These could be arranged in a format that is 2x3 for example. To create subfigures, each with their own caption, load:

```
\usepackage{caption}
\usepackage{subcaption}
```

Then, within the `figure` environment, have some `subfigure` environments.

```
\begin{figure}[h!]
  \begin{subfigure}[b]{0.3\textwidth}
     \includegraphics[width=\textwidth]{myfig.jpg}
     \caption{Subfigure 1 name}
  \end{subfigure}
  \begin{subfigure}[b]{0.3\textwidth}
     \includegraphics[width=\textwidth]{myfig.jpg}
     \caption{Subfigure 1 name}
  \end{subfigure}
  \caption{This is a figure}
  \label{f:figurelabel}
\end{figure}
```

Use blank lines between subfigure environments to signify new row for the figure.

No blank line means the figure will stay on the same row.

Maths

The equation handling within LaTeXis particularly powerful, and can handle the writing of as complicated an equation as it is possible to write. We will consider the basics of equations here. More detail can be found at the references at the end of this document. We must first load the `amsmath` package:

```
\usepackage{amsmath}
```

To create an equation, we create the `equation` environment, which can be labelled so we can reference in the text:

```
\begin{equation}
  \label{e:equ_enviro}
\end{equation}
```

The formatting of equations is as they are written, so there is no need to write with * or / unless they are wanted in the text. A simple example is:

```
y=a+b
```

$$y = a + b \tag{1}$$

Maths

Subscripts and superscripts:

```
f(x)=a_{0}+a_{1}x+a_{2}x^{2}+\dots
```

$$f(x) = a_0 + a_1x + a_2x^2 + \dots \tag{2}$$

For Greek letters, use `\<lettername>`:

```
\alpha, \beta, \gamma, \Gamma, \pi, \Pi, \
  phi
```

$$\alpha, \beta, \gamma, \Gamma, \pi, \Pi, \phi \tag{3}$$

For bold math font, so when using vectors for example:

```
\mathbf{a} \cdot \mathbf{b}
= |\mathbf{a}||\mathbf{b}|\cos \theta
```

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos \theta \tag{4}$$

Fractions:

```
R = \frac{\rho V L}{\mu}
```

$$R = \frac{\rho V L}{\mu} \tag{5}$$

Maths

Brackets around large equations ⇒ use `\left(` and `\right)`:

```
\frac{p_{2}}{p_{1}} =
\left(\frac{V_1}{V_2}^{\gamma}\right)
```

$$\frac{p_2}{p_1} = \left(\frac{V_1}{V_2}\right)^{\gamma} \tag{6}$$

Summation and integration:

```
\int_{a}^{b} f(x) dx \approx
\sum_{i=1}^{N} \Delta x f(x_i)
```

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} \Delta x f(x_i) \tag{7}$$

We can write our equations in the line of the text by placing the equation between $ $:

```
To obtain the distance between points $x$
and $y$, we can do $\sqrt{x^2+y^2}$
```

To obtain the distance between points x and y, we can do $\sqrt{x^2+y^2}$

Producing the Final Document

When we have written our document, we must compiler it. If we are using Texmaker, then this has built-in compiler buttons. Otherwise, the compiler must be called from command-line. There are two options for producing a .pdf file:

- If figures are .eps: `latex > dvips > ps2pdf`
- Else: `pdflatex`

NOTE: PDF means portable document format. This is the only document format you should distribute final drafts of documents with.

Extra Documentation

We really have only scratches the surface of LaTeX, and introduced the basic concepts. This lecture was about introducing you to another way of producing professional looking reports and technical documents. For further detail of the concepts introduced here you can look at the sites below.

LaTeXhas an online wiki, with considerable detail:

`https://en.wikibooks.org/wiki/LaTeX`

There is also a comprehensive set of lecture notes written and distributed by a lecturer as Washington Jefferson:

`http://www2.washjeff.edu/users/rhigginbottom/latex/lectures.html`

Summary

- Have introduced software for producing professional scientific reports: LaTeX
- Looked at how to:
  - Do basic document formatting
  - Write documents
  - Import figures and tables
  - Write equations