How to write in LaTeX

Lecture Note in LaTeX

Rakesh Jana

Department of Mathematics

Indian Institute of Technology Guwahati

Assam, India

IEEE Student Branch

IIT Guwahati

The materials of this book taken from various online source and books. This book created for education purpose only to learn LaTeX which author used to give talks on LaTeX . This book may contains many errors. Please inform author if you found so.

Last Updated on, August 23, 2018

Contents

1 Introduction 5

1.1 What is TeX? 5

1.2 What is LaTeX? 5

1.3 Prerequisites 6

2 LaTeX Basics 7

2.1 Preamble of Document 7

2.1.1 Package option and Special character 8

2.2 Text Formatting and Unit 9

2.2.1 Font Size 10

2.2.2 Font typefaces 11

2.3 Spacing and Paragraphs 11

2.3.1 Spaces 11

2.3.2 Line Break 11

2.3.3 Blank Spaces 12

2.3.4 Paragraph 13

2.3.5 Units 15

3 Tables and Figures 16

3.1 Table in LaTeX 16

3.1.1 Table with fixed length 18

3.1.2 Combining rows and columns 18

3.1.3 Multi-page tables 20

3.1.4 Positioning, Caption and Labeling 21

3.1.5 Miscellaneous 23

3.1.6 Reference 25

3.1.7 Table generator 26

3.2 Figure **26**
3.2.1 File Format . . . 26
3.2.2 `includegraphics` . . . 27
3.2.3 Positioning and Caption . . . 29
3.2.4 Wrap Figure . . . 31
3.2.5 Multiple images in one figure . . . 32
3.2.6 Wrap Table . . . 32

4 Mathematical Expression . . . **34**
4.1 Basic Expression **34**
4.1.1 Alignment . . . 35
4.2 Label and Tag **37**
4.3 Exponent and Brackets **39**
4.4 Matrix **42**
4.5 Spacing **43**
4.6 Chemical Equation **43**

5 Basic Drawing in LaTeX . . . **45**
5.1 Introduction **45**
5.2 Basic Drawing **46**
5.3 Geometric Figure **47**
5.4 `pgfplot` Package **50**
5.4.1 Graph Formulas . . . 51
5.4.2 3D Plot . . . 53
5.4.3 parametric Plot . . . 54
5.4.4 Plotting From data . . . 55
5.4.5 Bar Graphs . . . 56

6 Flowchart and Algorithm . . . **58**
6.1 Algorithm **58**
6.1.1 `algorithm2e` Package . . . 58
6.1.2 `algorithmicx` Package . . . 60
6.1.3 Listings package . . . 62
6.2 Flowchart **66**
6.2.1 Tikzstyle Command . . . 67
6.2.2 Nodes . . . 67
6.2.3 Arrow . . . 68
6.2.4 Example I . . . 68
6.2.5 Text width . . . 70
6.2.6 Example II . . . 71
6.3 Smartdiagram Package **71**
6.3.1 Customize smart diagram . . . 74

1. Introduction

1.1 What is TEX?

TEX (pronounced “Tech”, with “ch”) is a markup language created by Donald Knuth to typeset documents attractively and consistently. It’s also a Turing-complete programming language, in the sense that it supports the if-else construct, it can calculate (the calculations are performed while compiling the document), etc., but you would find it very hard to make anything else but typesetting with it. The fine control TeX makes it very powerful, but also difficult and time-consuming to use.

1.2 What is LATEX?

LATEX (pronounced either “Lah-tech") is a macro package based on TEX created by *Leslie Lamport*. Its purpose is to simplify TEX typesetting, especially for documents containing mathematical formula.

LATEX document processing is essentially programming. You create a text file in LATEXmarkup. The LATEX macro reads this to produce the final document. Clearly this has disadvantages in comparison with a WYSIWYG (What You See Is What You Get) program such as Microsoft Word:

- You can’t see the final result straight away.
- You need to know the necessary commands for LaTeX markup.
- It can sometimes be difficult to obtain a certain ’look’.

On the other hand, there are certain advantages to the markup language approach:

- The layout, fonts, tables and so on are consistent throughout.
- Mathematical formula can be easily typeset.
- Indices, footnotes, table of contents and references are generated easily.
- Mainly your documents will be correctly structured.

The LaTeX-like approach can be called WYSIWYM, i.e. What You See Is What You Mean: you can't see how the final version will look like while typing. Instead you see the logical structure of the document. LaTeX takes care of the formatting for you.

The LaTeX document is a plain text file containing the content of the document, with additional markup. When the source file is processed by the macro package, it can produce documents in several formats. LaTeX supports natively DVI and PDF, but using other software you can easily create PostScript, PNG, JPG, etc.

1.3 Prerequisites

LaTeX is a very easy system to learn, and requires no specialist knowledge. At a minimum, you'll need the following programs to edit LaTeX.

- An editor (You can use a basic text editor like notepad, but a dedicated LaTeX editor will be more useful).
  - On Windows and Unix, we commonly used TeXMaker and TeXstudio.
- The LaTeX binaries and style sheets - e.g. MiKTeX of TeXLive for Windows, teTeX for Unix/Linux and teTeX for Mac OS X

Here are the main programs you expect to
find in any (La)TeX distribution:

- **tex**: the simplest compiler: generates DVI from TeX source
- **pdftex**: generates PDF from TeX source
- **latex**: generates DVI from LaTeX source (the most used one)
- **pdflatex**: generates PDF from LaTeX source
- **dvi2ps**: converts DVI to PostScript
- **dvipdf**: converts DVI to PDF

When LaTeX was created, the only format it could create was DVI; then the PDF support was added by *pdflatex*, even if several people still don't use it. DVI is an old format, and it does not support hyperlinks for example, while PDF does, so passing through DVI you will bring all the bad points of that format to PDF.

The following diagram shows the relationships between the (La)TeX source code and all the formats you can create from it:

2. LaTeX Basics

A minimal example looks something like the following (the commands will be explained later):

```
\documentclass{article}

\begin{document}
        First document. This is a simple example, with no
        extra parameters or packages included.
\end{document}
```

The first line of code declares the type of document, in this case is a *article*. Then enclosed in the `\begin{document}` `\end{document}` tags you must write the text of your document.

2.1 Preamble of Document

The part of your .tex file before `\begin{document}` command is called the **preamble**. In the preamble you define the type of document you are writing, the language and several other elements. For instance, a normal document preamble would look like this:

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}

\title{First document}
\author{Rakesh Jana \thanks{Thanks to IEEE team}}
\date{June 2018}
```

Below a detailed description of each line:

- `\documentclass[12pt, a4paper]{article}` This defines the type of document. Some additional parameters inside brackets and comma-separated can be passed to the

command. In this example, the extra parameters set the font size (12pt) and the paper size (a4paper).

- `\usepackage[utf8]{inputenc}` This is the encoding for the document. Can be omitted or changed to another encoding but utf-8 is recommended. Unless you specifically need another encoding, or if you are unsure about it, add this line to the preamble.

The next three line define titles of your document. To display the title of your document you have to declare its components in the preamble, like we did and then you have to call them inside `\begin{document}` by the command `\maketitle`. A complete example is shown below.

```
\documentclass[12pt, a4paper]{article}
\usepackage[utf8]{inputenc}

\title{First document}
\author{Rakesh Jana \thanks{Thanks to IEEE team}}
\date{June 2018}

\begin{document}

        \begin{titlepage}
                \maketitle
        \end{titlepage}

        In this document some extra packages and parameters were added.
        There is some option packages are used,
        a encoding package, a pagesize and fontsize parameters.
\end{document}
```

- `\title{First document}` This is the title.
- `\author{Rakesh Jana}` Here you put the name of the Author(s) and, as a optional parameter, you can add the next command:
- `\thanks{Thanks to IEEE team}` This can be added after the name of the author, inside the braces of the title command. It will add a superscript and a footnote with the text inside the braces. Useful if you need to thank an institution in your article.
- `\date{June 2018}` You can enter the date manually or use the command `\today` so the date will be updated automatically at the time you compile your document.
- `\begin{titlepage} \end{titlepage}` Whatever you include in this `titlepage` environment will appear in the first page of your document.
- `\maketitle` This command will print the title, the author and the date in the format shown in the example. If it's not enclosed in a `titlepage` environment, it will be shown at the beginning of the document, above the first line.

2.1.1 Package option and Special character

Following document types available in the `class` option in `\documentclass{class}` command.

article	For short documents and journal articles. Is the most commonly used. No \part,\chapter divisions.
book	Default is two-sided.
report	For longer documents and dissertations. No \part divisions.
letter	Letter (?).
slides	For slides, rarely used. Large sans-serif font.
beamer	Slides in the Beamer class format. To make slide.

The most common options for the standard document classes are listed in following table:. Usage: \documentclass[*opt,opt*]{*class*}.

10pt/11pt/12pt	Font size.
letterpaper/a4paper	Paper size.
twocolumn	Use two columns.
twoside	Set margins for two-sided.
landscape	Landscape orientation.
draft	Double-space lines.

For example, if you want a report to be in 12pt type on A4, but printed one-sided in draft mode, you would use:

```
\documentclass[12pt,a4paper,oneside,draft]{report}
```

The following symbol characters are reserved by LATEX because they introduce a command and have a special meaning. These symbols and can be printed with special commands (in some cases - inside mathematical environment).

Character	Function	How to print it
#	Macro parameter	\#
$	Math mode	\$
%	Comment	\%
^	Superscript (in math mode)	\^{} or $\textasciicircum$
&	Separate column entries in tables	\&
_	Subscript (in math mode)	_
{ }	Processing block	\{ \}
~	Unbreakable space, use it whenever you want to leave a space which is unbreakable	$\textasciitilde$ or \~{}
\	Starting commands, which extend until the first non-alphanumerical character	$\textbackslash$ or $\backslash$

2.2 Text Formatting and Unit

In this part three basic text formatting tools will be explained: italics, bold and underline. Let's begin with an example:

```
Some of the \textbf{greatest} discoveries in \underline{science} were made by
students of \textit{Indian} \textsc{Institute} of Techonlogy, \textrm{Guwahati}.
 You can also use short command. {\bf \em This whole text in bold format}
```

Some of the **greatest** discoveries in science were made by students of *Indian* Institute of Techonlogy, Guwahati. You can also use short command. ***This whole text in bold format***

As you can see, there are three basic commands and they can be nested to get combined effects.

Command	*Declaration*	*Effect*
\textrm{*text*}	{\rmfamily *text*}	Roman family
\textsf{*text*}	{\sffamily *text*}	Sans serif family
\texttt{*text*}	{\ttfamily *text*}	`Typewriter family`
\textmd{*text*}	{\mdseries *text*}	Medium series
\textbf{*text*}	{\bfseries *text*}	**Bold series**
\textup{*text*}	{\upshape *text*}	Upright shape
\textit{*text*}	{\itshape *text*}	*Italic shape*
\textsl{*text*}	{\slshape *text*}	*Slanted shape*
\textsc{*text*}	{\scshape *text*}	Small Caps shape
\emph{*text*}	{\em *text*}	*Emphasized*
\textnormal{*text*}	{\normalfont *text*}	Document font
\underline{*text*}		Underline

The commands \it and \bf also work to *italicize* and **boldface** text, but it's not recommended to use them since they don't preserve previous styles.

2.2.1 Font Size

Font sizes are identified by special names, the actual size is not absolute but relative to the font size declared in the \documentclass statement.

```
Here {\huge huge font size} is set and  the {\footnotesize Foot note size also}.
This  {\large large} set of font size  is {\Large larger} then previous one.
```

Here huge font size is set and the Foot note size also. This large set of font size is larger then previous one.

here is the complete reference guide.

\tiny	tiny
\scriptsize	scriptsize
\footnotesize	footnotesize
\small	small
\normalsize	normalsize
\large	large
\Large	Large
\LARGE	LARGE
\huge	huge
\Huge	Huge

2.2.2 Font typefaces

In LaTeXthe default font typeface is the **Computer Modern family**. You can change this font typeface for another that better suits your style. You need to use `\usepackage{tgbonum}` in your preamble. It establishes the font family **TeXGyre Bonum**, whose font package name is *tgbonum*, as the default font for this document. For example

```
Let us try different text style.
{\fontfamily{qcr}\selectfont This text uses a different font typeface}
```

Let us try different text style. `This text uses a different font typeface`

The command `\fontfamily{qcr}\selectfont` will set the **TEX gyre cursor font typeface**, whose fontcode is `qcr`, for the text inside the braces. Please see the below link for other type of fonts. http://www.tug.dk/FontCatalogue/

2.3 Spacing and Paragraphs

In this part we explained how to format paragraphs, change the text alignment and insert blank spaces.

2.3.1 Spaces

As you might notice "Whitespace" characters, such as blank or tab, are treated uniformly as "space" by LaTeX. Several consecutive whitespace characters are treated as one "space".

An empty line between two lines of text defines the end of a paragraph. Several empty lines are treated the same as one empty line. The text below is an example. Try the following code and see the differences.

```
It does not matter whether you
enter one or several       spaces
after a word.
An empty line starts a new
paragraph.
```

It does not matter whether you enter one or several spaces after a word. An empty line starts a new paragraph.

2.3.2 Line Break

As mentioned before, there's more than one way to insert line breaks. First look into following example.

```
Something in this document. This paragraph contains no information  and its
purposes is to provide an example on how to insert white  spaces
and lines breaks.\\ When a line break is inserted, the text is not indented,
 there  are a couple of extra commands do line breaks. \newline This paragraph
 provides no information whatsoever. We are exploring
line breaks.  \hfill \break And combining two commands
```

Something in this document. This paragraph contains no information and its purposes is to provide an example on how to insert white spaces and lines breaks. When a line break is inserted, the text is not indented, there are a couple of extra commands do line breaks.
This paragraph provides no information whatsoever. We are exploring line breaks. And combining two commands

The following commands are use for line break.

Character	Function
\\	Breaks the line at the point of the command
\newline	This command work same as \\
*	Breaks the line at the point of the command and additionally prohibits a page break after the forced line break.
\break	Breaks the line without filling the current line. This will result in very bad formatting if you do not fill the line yourself.
\hfill\break	This will produce the same result as \newline and \\..
\linebreak[number]	It breaks the line at the point of the command. The number provided as an argument represents the priority of the command in a range of 0 to 4. (0 means it will be easily ignored and 4 means do it anyway).

2.3.3 Blank Spaces

There are two commands to insert page breaks, \clearpage and newpage.

If the command \clearpage is used, and there are stacked floating elements, such as tables or figures, they will be flushed out before starting the new page. In the example above the same image is inserted three times. Since the page break is inserted before all the figures are displayed, remaining images are inserted in an empty page before continuing with the text below the brake point.

If this is not what you need, you can use \newpage instead.

Horizontal and vertical spaces of arbitrary length may be inserted with \hspace and \vspace.

```
Horizontal \hspace{1cm} spaces can be inserted manually. Useful to control
the fine-tuning in the layout of pictures.

Left Side \hfill Right Side

5mm vertical space can be inserted manually.  This is my top text

\vspace{5mm} %5mm vertical space

This text still at the top, 5mm below the first paragraph.

\vfill

Text at the bottom of the page.
```

Horizontal spaces can be inserted manually. Useful to control the fine-tuning in the layout of pictures.
Left Side Right Side
5mm vertical space can be inserted manually. This is my top text

This text still at the top, 5mm below the first paragraph.
Text at the bottom of the page.

- `\hspace{1cm}` Inserts a horizontal space whose length is 1cm. Other LaTeXunits can be used with this command.
- `\hfill` Inserts a blank space that will stretch accordingly to fill the space available in that line. The commands `\hrulefill` and `\dotfill` do the same as `\hfill` but instead of blank spaces they insert a horizontal ruler and a string of dots, respectively.
- `\vspace{5mm}` Inserts a vertical spaces whose length is 5mm. Other LaTeXunits can be used with this command.
- `\vfill` Inserts a blank space that will stretch accordingly to fill the vertical space available in that page.
- `\smallskip` Adds a 3pt space plus or minus 1pt depending on other factors (document type, available space, etc)
- `\medskip` Adds a 6pt space plus or minus 2pt depending on other factors (document type, available space, etc)
- `\bigskip` Adds a 12pt space plus or minus 4pt depending on other factors (document type, available space, etc)

2.3.4 Paragraph

To start a new paragraph in LaTeX, as said before, you must leave a blank line in between. There's another way to start a new paragraph, look at the following code snippet.

```
This is the text in first paragraph. This is the text in first
paragraph.
 This is the text in first paragraph. \par This is the text in second paragraph.
 This is the text in second  paragraph. This is the text in second paragraph.
```

This is the text in first paragraph. This is the text in first paragraph. This is the text in first paragraph.

This is the text in second paragraph. This is the text in second paragraph. This is the text in second paragraph.

Paragraphs in LaTeX are fully justified, i.e. flush with both the left and right margins. If you would like to change the justification of a paragraph, LaTeX has the following three environments: **center, flushleft and flushright**. For example

```
\begin{center}
        This text is in the center. \\ Whatever you will write it will
        automatically center justify.
\end{center}
\begin{flushright}
        This text is in the right. \\ Whatever you will write it will
        automatically right justify.
\end{flushright}
```

This text is in the center.
Whatever you will write it will automatically center justify.
This text is in the right.
Whatever you will write it will automatically right justify.

Environment	*Declaration*	Justify
\begin{center}	\centering	center-justifies the paragraph
\begin{flushleft}	\raggedright	right-justifies the paragraph
\begin{flushright}	\raggedleft	left-justifies the paragraph

By default, LaTeX does not indent the first paragraph of a section but it does from subsequent paragraph. The size of the subsequent paragraph indents is determined by the parameter \parindent. By default \parindent define by 6 pt (that is, one tab). But we can redefine it as our choice whenever we want.

```
\setlength{\parindent}{10ex} This is the text in first paragraph.
This is the text in first paragraph.  This is the text in first paragraph. \par
This is the text in second paragraph. This is the text in second  paragraph.
This is the text in second paragraph. \par \noindent  This is the text in third
paragraph. This is the text in third paragraph. This is the text in third paragraph.
```

This is the text in first paragraph. This is the text in first paragraph. This is the text in first paragraph.

This is the text in second paragraph. This is the text in second paragraph. This is the text in second paragraph.
This is the text in third paragraph. This is the text in third paragraph. This is the text in third paragraph.

The default length of this parameter is set by the document class used. It is possible to change the indent size of the paragraph by using the command \setlength.

- \setlength{\parindent}{10ex} will be indented !10ex! (an "ex" equals the lenght of the "x" in the current font)
- \noindent it create a non-indented paragraph, the command \noindent need to use at the beginning of the paragraph.

If you want to indent a paragraph that is not indented you can use \indent above it. It should be noted that this command will only have an effect when \parindent is not set to zero.

2.3.5 Units

Below a description of available units in LaTeX.

Abbreviation	Value
pt	a point is approximately 1/72.27 inch, that means about 0.0138 inch or 0.3515 mm (exactly point is defined as 1/864 of American printers foot that is 249/250 of English foot)
mm	a millimeter
cm	a centimeter
in	inch
ex	roughly the height of an 'x' (lowercase) in the current font (it depends on the font used)
em	roughly the width of an 'M' (uppercase) in the current font (it depends on the font used)
mu	math unit equal to 1/18 em, where em is taken from the math symbols family

Lengths are units of distance relative to some document elements. Lengths can be changed by the command as we have changed it in earlier section:

```
\setlength{\lengthname}{value_in_specified_unit}
```

For example, in a `two-column` document the column separation can be set to 1 inch by: `\setlength{\columnsep}{1in}`

Below is a table with some of the most common lengths and their description which we can changed.

Length	Description
`\baselineskip`	Vertical distance between lines in a paragraph
`\columnsep`	Distance between columns
`\columnwidth`	The width of a column
`\evensidemargin`	Margin of even pages, commonly used in two-sided documents such as books
`\linewidth`	Width of the line in the current environment.
`\oddsidemargin`	Margin of odd pages, commonly used in two-sided documents such as books
`\paperwidth`	Width of the page
`\paperheight`	Height of the page
`\parindent`	Paragraph indentation
`\parskip`	Vertical space between paragraphs
`\tabcolsep`	Separation between columns in a table (tabular environment)
`\textheight`	Height of the text area in the page
`\textwidth`	Width of the text area in the page
`\topmargin`	Length of the top margin

3. Tables and Figures

Tables and Images are common but essential elements in most scientific documents, LaTeX provides a large set of tools to customize tables and to handle images and make them look exactly what you need. In this chapter we will discuss first how to customize tables, change the size, combine cells, change the colour of cells and so on and then we explained how to include images in the most common formats, how to shrink, enlarge and rotate them, and how to reference them within your document.

3.1 Table in LaTeX

The `tabular` environment is the default LaTeX method to create tables. You must specify a parameter to this environment, `{c c c}` tells LaTeX that there will be three columns and that the text inside each one of them must be centred.

Tables can be created using tabular environment.

```
\begin{tabular}[pos]{cols}
        table content
\end{tabular}
```

Let us look into following example.

```
\begin{center}
        \begin{tabular}{ c c c }
        a   & bc   & def \\
        gh  & ijk  & lmno \\
        pqr & stuv & wxyzq
        \end{tabular}
\end{center}
```

a	bc	def
gh	ijk	lmno
pqr	stuv	wxyzq

It was already said that the `tabular` environment is used to type tables. We use `\begin{center}` `\end{center}` to make table in center of the page, otherwise by default it will appear in left side of the page. To be more clear about how it works below is a description of each command.

- `{ c c c }`
  It's represent the table have three columns. Each `c` means that the contents of the column will be centred, you can also use `r` to align the text to the right and `l` for left alignment.
  - `{ |c | c | c | }`
    This declares that three columns, separated by a vertical line, are going to be used in the table.

- `\hline`
  This will insert a horizontal line on top of the table and at the bottom too. There is no restriction on the number of times you can use `\hline`.

- `a & bc & def`
  Each `&` is a cell separator and the double-backslash `\\` sets the end of this row.

```
\begin{center}
        \begin{tabular}{ |c|c|c| }
                \hline
                cell1 & cell2 & cell3 \\
                cell4 & cell5 & cell6 \\
                cell7 & cell8 & cell9 \\
                \hline
        \end{tabular}
\end{center}
```

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

Following example shows double vertical and horizontal lines, when properly used help to keep the information within the table well organized.

```
\begin{center}
        \begin{tabular}{||c c c c||}
                \hline
                Col1 & Col2 & Col2 & Col3 \\ [0.5ex]
                \hline\hline
                1 & 6 & 87837 & 787 \\
                \hline
                2 & 7 & 78 & 5415 \\
                \hline
                3 & 545 & 778 & 7507 \\
                \hline
                4 & 545 & 18744 & 7560 \\
                \hline
                5 & 88 & 788 & 6344 \\ [1ex]
                \hline
        \end{tabular}
\end{center}
```

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

using `\\[0.5ex]` we can specify space between consecutive two rows.

3.1.1 Table with fixed length

When formatting a table you might require a fixed length either for each column or for the entire table. For this we must import the package array in the preamble of your LATEX file with the next command `\usepackage{array}`.

```
\begin{center}
\begin{tabular}{ | m{5em} | m{1cm}| m{1cm} | }

\hline
cell1 dummy text dummy text dummy text &
                        cell2 & cell3 \\ \hline
cell1 dummy text dummy text dummy text &
                        cell5 & cell6 \\ \hline
cell7 & cell8 & cell9 \\ \hline

\end{tabular}
\end{center}
```

cell1 dummy text dummy text dummy text	cell2	cell3
cell1 dummy text dummy text dummy text	cell5	cell6
cell7	cell8	cell9

In the `tabular` environment, the parameter `m{5em}` sets a length of $5em$ for first column ($1cm$ for the other two) and centres the text in the middle of the cell. The aligning options are `m` for middle, `p` for top and `b` for bottom. In standard tables new lines must be inserted manually so the table won't stretch out of the text area, when using this parameters the text is automatically formatted to fit inside each cell.

3.1.2 Combining rows and columns

Rows and columns can be combined in a bigger cell. The example below is an example of the `\multicolumn` command to combine columns. To combine rows the package `multirow` must be imported with `\usepackage{multirow}` in your preamble, then you can use the `\multirow` command in your document:

```
\begin{center}
        \begin{tabular}{ |p{3cm}||p{3cm}|p{3cm}|p{3cm}|  }
        \hline
        \multicolumn{4}{|c|}{Country List} \\\hline
        Country Name or Area Name & ISO ALPHA 2 Code
                        & ISO ALPHA 3 Code & ISO numeric Code \\ \hline
        Afghanistan   & AF & AFG & 004 \\
        Aland Islands & AX & ALA & 248 \\
```

```
        Albania       & AL & ALB & 008 \\
        Algeria       & DZ & DZA & 012 \\
        American Samoa& AS & ASM & 016 \\
        Andorra       & AD & AND & 020 \\
        Angola        & AO & AGO & 024 \\ \hline
        \end{tabular}
\end{center}
```

Country List			
Country Name or Area Name	ISO ALPHA 2 Code	ISO ALPHA 3 Code	ISO numeric Code
Afghanistan	AF	AFG	004
Aland Islands	AX	ALA	248
Albania	AL	ALB	008
Algeria	DZ	DZA	012
American Samoa	AS	ASM	016
Andorra	AD	AND	020
Angola	AO	AGO	024

Let's see each part of the command `\multicolumn{4}{|c|}{Country List}\\`:

- `{4}`
  The number of columns to be combined, 4 in this case.
- `{|c|}`
  Delimiters and alignment of the resulting cell, in this case the text will be centred and a vertical line will be drawn at each side of the cell. `{Country List}`
  This text to be displayed inside the cell.

```
\begin{tabular}{ |1|1|1| }
        \hline
        \multicolumn{3}{ |c| }{Team sheet} \\
        \hline
        Goalkeeper & GK & Paul Robinson \\
        \hline
        \multirow{4}{*}{Defenders} & LB & Lucas Radebe \\
        & DC & Michael Duburry \\
        & DC & Dominic Matteo \\
        & RB & Didier Domi \\
        \hline
        \multirow{3}{*}{Midfielders} & MC & David Batty \\
        & MC & Eirik Bakke \\
        & MC & Jody Morris \\
        \hline
        Forward & FW & Jamie McMaster \\
        \hline
        \multirow{2}{*}{Strikers} & ST & Alan Smith \\
        & ST & Mark Viduka \\
        \hline
\end{tabular}
```

Team sheet		
Goalkeeper	GK	Paul Robinson
Defenders	LB	Lucas Radebe
	DC	Michael Duburry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

Let's see each part of the command `\multirow{4}{*}{Defenders}`:

- `{4}`
  The number of rows to be combined, 4 in this case.
- `{*}`
  The width of the column, in this case unspecified.
- `{Defenders}`
  This text to be displayed inside the cell.

Let us consider `\multirow{3}{4em}{Multiple row}`. The first one is the number of rows to be combined, 3 in the example. The second parameter is the width of the column, $4em$ in the example. Finally, the third parameter is the content of the cell.

3.1.3 Multi-page tables

If you have to insert a very long table, which takes up two or more pages in your document, use the `longtable` package. First, add to the preamble the line `\usepackage{longtable}` This will make the command `longtable` available.

```
 \begin{longtable}[c]{| c | c |}
        \caption{Long table caption.\label{long}}\\

\hline
\multicolumn{2}{| c |}{Begin of Table}\\
\hline
Table Header & Table Header \\
\hline
\endfirsthead

\hline
\multicolumn{2}{|c|}{Continuation of Table \ref{long}}\\
\hline
Table Header & Table Header \\
\hline
\endhead
\hline \hline
```

```
Table Footer & Table Footer \\
\hline

\endfoot

\hline
\multicolumn{2}{| c |}{End of Table}\\
\hline\hline
\endlastfoot

        Lots of lines & like this\\
        Lots of lines & like this\\
        Lots of lines & like this\\
        Lots of lines & like this\\
        Lots of lines & like this\\
        Lots of lines & like this\\
        Lots of lines & like this\\
        Lots of lines & like this\\
        ...
        Lots of lines & like this\\
\end{longtable}
```

Compile the above the code by your own but before that do not forget to add `\usepackage{longtable}` in preamble and replace `...` with 100 of `Lots of lines & like this\\` this lines such that your table take two pages.

`longtable` behaviour is similar to the default `tabular`, but generates tables that can be broken by the standard LaTeX page-breaking algorithm. There are four elements long-table specific.

- `\endfirsthead`
  Everything above this command will appear at the beginning of the table, in the first page.
- `\endhead`
  Whatever you put before this command and below `\endfirsthead` will be displayed at the top of the table in every page except the first one.
- `\endfoot`
  Similar to `\endhead`, what you put after `\endhead` and before this command will appear at the bottom of the table in every page except the last one.
- `\endlastfoot`
  Similar to `\endfisthead`. The elements after `\endfoot` and before this command will be displayed at the bottom of the table but only in the last page where the table appears.

3.1.4 Positioning, Caption and Labeling

Positioning a table is easy if they're inside a float `table` environment. LaTeX automatically float your table in appropriate place. Let us look into this example. Before that for this we need to add `\usepackage{float}` in your preamble.

```
\begin{table}[h!]
        \centering
```

```
        \begin{tabular}{||c c c c||}
                \hline
                Col1 & Col2 & Col2 & Col3 \\ [0.5ex]
                \hline\hline
                1 & 6 & 87837 & 787 \\
                2 & 7 & 78 & 5415 \\
                3 & 545 & 778 & 7507 \\
                4 & 545 & 18744 & 7560 \\
                5 & 88 & 788 & 6344 \\ [1ex]
                \hline
        \end{tabular}
\end{table}
```

In this example the commands.: `\centering` use to make centres the table relative to the float container element.

This environment uses a positioning parameter passed inside brackets, it can take the next values:

Parameter	Position
h	Place the float here. Will place the table here approximately.
t	Position the table at the top of the page.
b	Position the table at the bottom of the page.
p	Put the table in a special page, for tables only.
!	Auto detect "good" float positions. Override internal LaTeX parameters.
H	Place the table at this precise location, pretty much like `h!`

Tables can be captioned, labelled and referenced if it is inside `table` environment.

```
\begin{table}[h!]
        \centering
        \begin{tabular}{||c c c c||}
                \hline
                Col1 & Col2 & Col2 & Col3 \\ [0.5ex]
                \hline\hline
                1 & 6 & 87837 & 787 \\
                2 & 7 & 78 & 5415 \\
                3 & 545 & 778 & 7507 \\
                4 & 545 & 18744 & 7560 \\
                5 & 88 & 788 & 6344 \\ [1ex]
                \hline
        \end{tabular}
        \caption{Table to test captions and labels}
        \label{table:1}
\end{table}
The table \ref{table:1} is an example of referenced \LaTeX\ elements.
```

The table 3.1 is an example of referenced LaTeX elements.

There are three important commands in the example:

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Table 3.1: Table to test captions and labels

- `\caption{Table to test captions and labels}`
  As you may expect this command sets the caption for the table, if you create a list of tables this caption will be used there. You can place it above or below the table.
- `\label{table:1}`
  If you need to refer the table within your document, set a label with this command. The label will number the table, and combined with the next command will allow you to reference it.
- `\ref{table:1}`
  This code will be substituted by the number corresponding to the referenced table.

To create a list of tables is straightforward.

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\begin{document}

        \listoftables

        ...
\end{document}
```

The caption of each table will be used to generate this list using the command `\listoftables`

3.1.5 Miscellaneous

Several table elements can be modified to achieve a good-looking document using `\setlength` command.

```
\setlength{\arrayrulewidth}{1mm}
\setlength{\tabcolsep}{18pt}
\renewcommand{\arraystretch}{1.5}
        \begin{tabular}{ |p{3cm}|p{3cm}|p{3cm}|  }
                \hline
                \multicolumn{3}{|c|}{Country List} \\
                \hline
                Country Name     or Area Name& ISO ALPHA 2 Code &ISO ALPHA 3 \\
                \hline
                Afghanistan & AF &AFG \\
```

```
            Aland Islands & AX   & ALA \\
            Albania &AL & ALB \\
            Algeria    &DZ & DZA \\
            American Samoa & AS & ASM \\
            Andorra & AD & AND   \\
            Angola & AO & AGO \\
            \hline
        \end{tabular}
```

Country List		
Country Name or Area Name	ISO ALPHA 2 Code	ISO ALPHA 3
Afghanistan	AF	AFG
Aland Islands	AX	ALA
Albania	AL	ALB
Algeria	DZ	DZA
American Samoa	AS	ASM
Andorra	AD	AND
Angola	AO	AGO

There three major length are there in `\tabular` environment.

- `\setlegth{\arrayrulewidth}{1mm}`
  This sets the thickness of the borders of the table. In the example is 1mm but you can use other units..
- `\setlength{\tabcolsep}{18pt}`
  The space between the text and the left/right border of its containing cell is set to 18pt with this command.
- `\renewcommand{\arraystretch}{1.5}` The height of each row is set to 1.5 relative to its default height.

It is a common practice to use two colours for alternating rows in a tables to improve readability. This can be achieved in LaTeXwith the package `xcolor` and the table parameter. Include `\usepackage{xcolor}` in the preamble to use color in your document.

```
\setlength{\arrayrulewidth}{1mm}
\setlength{\tabcolsep}{18pt}
\renewcommand{\arraystretch}{1.5}
{\rowcolors{3}{gray!30}{green!70!yellow!40}

\begin{tabular}{ |p{3cm}|p{3cm}|p{3cm}|  }
```

```
\hline
        \multicolumn{3}{|c|}{Country List} \\
\hline
Country Name     or Area Name& ISO ALPHA 2 Code &ISO ALPHA 3 \\ \hline
        Afghanistan & AF &AFG \\
        Aland Islands & AX   & ALA \\
        Albania &AL & ALB \\
        Algeria    &DZ & DZA \\
        American Samoa & AS & ASM \\
        Andorra & AD & AND   \\
        Angola & AO & AGO \\ \hline
\end{tabular}
```

Country List		
Country Name or Area Name	ISO ALPHA 2 Code	ISO ALPHA 3
Afghanistan	AF	AFG
Aland Islands	AX	ALA
Albania	AL	ALB
Algeria	DZ	DZA
American Samoa	AS	ASM
Andorra	AD	AND
Angola	AO	AGO

3.1.6 Reference

Let us recall tables can be created using tabular environment.

```
        \begin{tabular}[pos]{cols}
                table content
        \end{tabular}
```

Available Vertical position (pos) option in `tabular` argument

Abbreviation	Definition
t	the line at the top is aligned with the text baseline
b	the line at the bottom is aligned with the text baseline
c	(Default option) table is centred to the text baseline

The `cols` : Defines the alignment and the borders of each column. It can have the following values:

Abbreviation	Definition
`l`	left-justified column
`c`	centred column
`r`	right-justified column
`p{width}`	paragraph column with text vertically aligned at the **top**
`m{width}`	paragraph column with text vertically aligned in the **middle** (requires `array` package)
`b{width}`	paragraph column with text vertically aligned at the **bottom** (requires `array` package)
`|`	vertical line
`||`	double vertical line
`*{num}{form}`	the format `form` is repeated `num` times; for example `*{3}{|l}|` is equal to `|l|l|l|`

To separate between cells and introducing new lines use the following commands:

Abbreviation	Definition
`&`	column separator
`\\`	start new row (additional space may be specified after `\\` using square brackets, such as `\\[6pt]`)
`\hline`	horizontal line between rows
`\cline{i-j}`	partial horizontal line beginning in `column i` and ending in `column j`

3.1.7 Table generator

Quickly create even complex LaTeX tables online with LaTeX table generator – cells merging is supported together with borders editing.
https://www.tablesgenerator.com/

3.2 Figure

Images are essential elements in most of the scientific documents. LaTeX provides several options to handle images and make them look exactly what you need. In this article is explained how to include images in the most common formats, how to shrink, enlarge and rotate them, and how to reference them within your document.

3.2.1 File Format

When producing a DVI file, a postscript graphics file like encapsulated postscript (EPS) is used. When using the `pdflatex` command, several graphics formats can be used: PNG, JPG, and PDF; however, an `EPS` file cannot be used directly. To use `eps` file you have to use `\usepackage{epstopdf}` which will automatically convert an `EPS` to a `PDF` file. If it does not automatically convert, an error message will appear.

3.2.2 includegraphics

The table environment and the caption package were discussed in earlier section. Essentially, the figure environment has the same relationship with figures as the table environment has with tables.

Figures are floated to optimize space, but you can use the same optional arguments for placement with or without the exclamation point: h, t, b, or p. The captions are automatically numbered and the caption package can be used to format the caption.

One of the following packages is needed for graphics: graphics or graphicx. The graphics driver selected will control how the graphic appears and is printed.

```
\begin{figure}[optional argument]
        \centering
        \includegraphics[optional arguments]{filename}
        \caption{caption text}
\end{figure}
```

There is special version of the figure environment, figure*, that behaves exactly like table*, that mens, there will be no numbering in caption.

Let us first discuss with only \includegraphics command. Suppose you have a image with name iitg.jpg in the directory images.

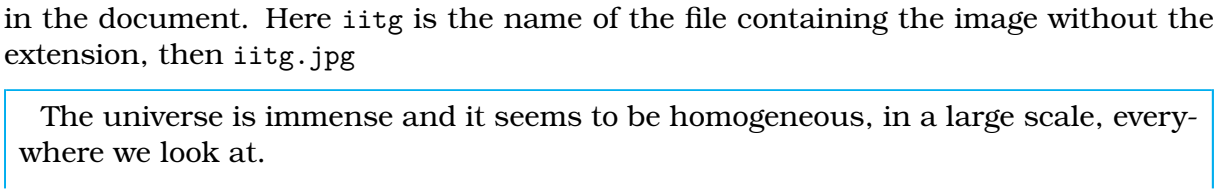
```
\documentclass{article}
\usepackage{graphicx}
\graphicspath{ {images/} }

\begin{document}
        The universe is immense and it seems to be homogeneous,
        in a large scale, everywhere we look at.

        \includegraphics{iitg}

        There's a picture of a galaxy above
\end{document}
```

Latex can not manage images by itself, so we need to use the graphicx package. To use it, we include the following line in the preamble: \usepackage{graphicx}

The command \graphicspath{ {images/} } tells LaTeX that the images are kept in a folder named images under the current directory.

The \includegraphics{iitg} command is the one that actually included the image in the document. Here iitg is the name of the file containing the image without the extension, then iitg.jpg

The universe is immense and it seems to be homogeneous, in a large scale, everywhere we look at.



There's a picture of a galaxy above

As you might notice that LaTeX does not automatically scale the images. You are required to scale your image manually.

```
\begin{center}
        \includegraphics[scale=0.5]{iitg.jpg}
\end{center}
```

The command `\includegraphics[scale=0.5]{iitg.jpg}` will include the image `iitg` in the document, the extra parameter `scale=0.5` will do exactly that, scale the image half of its real size.

You can also scale the image to a some specific width and height by using the following command `\includegraphics[width=3cm, height=4cm]{iitg}`

As you probably have guessed, the parameters inside the brackets `[width=3cm, height=4cm]` define the `width` and the `height` of the picture. You can use different units for these parameters. If only the width parameter is passed, the height will be scaled to keep the aspect ratio.

As we know the command `\textwidth` contain the width of current page size. We can use it to include our image. For example.

```
\includegraphics[width=\textwidth]{iitg}
```

Instead of `\textwidth` you can use any other default LaTeX length: `\columnsep`, `\linewidth`, `\textheight`, `\paperheight`, etc.

There is another common option when including a picture within your document, to **rotate** it. The parameter `angle=45` rotates the picture 45 degrees counter-clockwise. To rotate the picture clockwise use a negative number. For example

```
\includegraphics[scale=0.5, angle=45]{iitg}
```

3.2.3 Positioning and Caption

In the previous section was explained how to include images in your document, but the combination of text and images may not look as we expected. To change this we need to introduce a new environment `figure` which is same as `table`, as defined in earlier section. For example

```
\begin{figure}[t]
        \centering
        \includegraphics[width=8cm]{iitg}
        \caption{IIT Guwahati}\label{fig:iitg}
\end{figure}
As you can see in the figure \ref{fig:iiitg}, the  Indian
 Institute of Technology Guwahati is really beautiful.
```

As you can see in the figure 3.1, the Indian Institute of Technology Guwahati is really beautiful.

Figure 3.1: IIT Guwahati

Figures, just as many other elements in a LaTeX document (equations, tables, plots, etc) can be referenced within the text. This is very easy, just add a `label` to the figure environment just after `\caption`, then later use that label to refer the picture.

There are three commands that generate cross-references in this example.

- `\label{fig:iitg}`
  This will set a label for this figure. Since labels can be used in several types of elements within the document, it's a good practice to use a prefix, such as fig: in the example.
- `\ref{fig:iitg}`
  This command will insert the number assigned to the figure. It's automatically generated and will be updated if insert some other figure before the referenced one.
- `\pageref{fig:mesh1}`
  This prints out the page number where the referenced image appears.

The `\caption` is mandatory to reference a figure.

The additional command `\centering` will centre the picture. The default alignment is `left`.

Another great characteristic in a LaTeXdocument is the ability to automatically generate a list of figures same as list of table. This is straightforward.

```
\listoffigures
```

Also floating of figure is same as floating of table. This environment uses a positioning parameter passed inside brackets, it can take the next values:

Parameter	Position
h	Place the float here. Will place the table here approximately.
t	Position the table at the top of the page.
b	Position the table at the bottom of the page.
p	Put the table in a special page, for tables only.
!	Auto detect "good" float positions. Override internal LaTeX parameters.
H	Place the table at this precise location, pretty much like `h!`

You can put more than one value in the parameter, for instance, if you write `[ht]` LaTeX will try to position the figure here, but if it's not possible (the space may be insufficient)

then the figure will appear at the top of the page. It is recommended to use more than one positioning parameter to prevent unexpected results.

3.2.4 Wrap Figure

It's also possible to wrap the text around a figure. When the document contains small pictures this makes it look better. For this, you have to import the package `wrapfig`. Add to the preamble the line `\usepackage{wrapfig}`.

```
\begin{wrapfigure}{r}{0.25\textwidth} %this figure will be at the right
        \centering
        \includegraphics[width=0.25\textwidth]{images/iitg}
\end{wrapfigure}

There are several ways to plot a function of two variables,
depending on the information you are interested in. For
instance, if you want to see the mesh of a function so it
easier to see the derivative you can use a plot like the
one on the left.


\begin{wrapfigure}{l}{0.25\textwidth}
        \centering
        \includegraphics[width=0.25\textwidth]{iitg}
\end{wrapfigure}

On the other side, if you are only interested on
certain values you can use the contour plot, you
can use the contour plot, you can use the contour
plot, you can use the contour plot, you can use
the contour plot, you can use the contour plot,
you can use the contour plot, like the one on the left.

On the other side, if you are only interested on
certain values you can use the contour plot, you
can use the contour plot, you can use the contour
plot, you can use the contour plot, you can use the
contour plot, you can use the contour plot,
you can use the contour plot,
like the one on the left.
```

In the commands `\begin{wrapfigure}{l}{0.25\textwidth} \end{wrapfigure}` the part `{l}` defines the alignment of the figure. Set `l` for left and `r` for right. Furthermore, if you are using a book or any similar format, use instead `o` for the outer edge and `i` for the inner edge of the page.

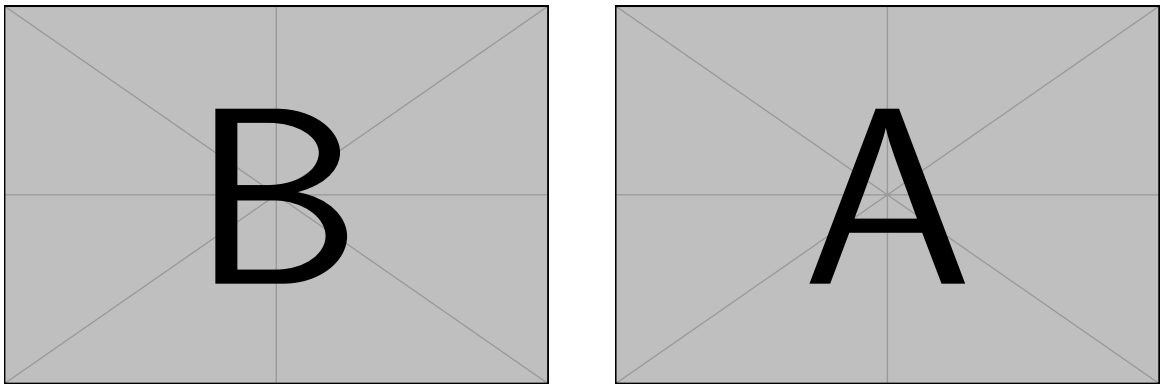
`{0.25\textwidth}` set the width of figure box. It's not the width of the image itself, that must be set in the `includegraphics` command. Notice that the length is relative to the text width, but normal units can also be used (cm, in, mm, etc).

`\centering` This was already explained, but in this example the image will be centred by using its container as reference, instead of the whole text.

3.2.5 Multiple images in one figure

It is possible to insert several images in one figure, each one with its own reference and label using `subfigure` environment but for this you you must import the package `subcaption` by adding to the preamble `\usepackage{subcaption}`

```
\begin{figure}[h]

        \begin{subfigure}{0.5\textwidth}
                \includegraphics[width=0.9\linewidth,
                 height=5cm]{example-image-a}
                \caption{Caption1}\label{fig:subim1}
        \end{subfigure}
        \begin{subfigure}{0.5\textwidth}
                \includegraphics[width=0.9\linewidth,
                 height=5cm]{example-image-b}
                \caption{Caption 2}\label{fig:subim2}
        \end{subfigure}

        \caption{Caption for this figure with two images}\label{fig:image2}
\end{figure}
```



(a) Caption1 (b) Caption 2

Figure 3.2: Caption for this figure with two images

3.2.6 Wrap Table

Wrapping text around a table is also possible same as wrapping figure around text. If your table don't take all available space and you want to put text next or before it, is possible with the package `wrapfig`. Try the following code.

```
        Lorem ipsum dolor sit amet, consectetuer
        adipiscing elit.Lorem ipsum dolor sit amet, consectetuer
        adipiscing elit.Lorem ipsum dolor sit amet, consectetuer
        adipiscing elit.
\begin{wraptable}{r}{8cm}
        \begin{tabular}{||c c c c||}
                \hline
                Col1 & Col2 & Col2 & Col3 \\ [0.5ex]
```

```
                \hline\hline
                1 & 6 & 87837 & 787 \\
                \hline
                2 & 7 & 78 & 5415 \\
                \hline
                3 & 545 & 778 & 7507 \\
                \hline
                4 & 545 & 18744 & 7560 \\
                \hline
                5 & 88 & 788 & 6344 \\ [1ex]
                \hline
        \end{tabular}
\caption{Table inside a floating element}
\label{table:ta}
\end{wraptable}
Lorem ipsum dolor sit amet, consectetuer
adipiscing elit.Lorem ipsum dolor sit amet, consectetuer
adipiscing elit.Lorem ipsum dolor sit amet, consectetuer
adipiscing elit.
```

4. Mathematical Expression

The feature that makes LaTeX the right edition tool for scientific documents is the ability to render complex mathematical expressions. This article explains the basic commands to display equations.

4.1 Basic Expression

Basic equations in LaTeX can be easily "programmed", for example:

```
We know  Pythagorean theorem $x^2 + y^2 = z^2$ was  proved to be invalid for
 other exponents, that is, following equation has no integer solutions:
$$ x^n + y^n = z^n, \quad \forall n\geq 3 $$
```

We know Pythagorean theorem $x^2 + y^2 = z^2$ was proved to be invalid for other exponents, that is, following equation has no integer solutions:

$$x^n + y^n = z^n, \quad \forall n \geq 3$$

As you see, the way the equations are displayed depends on the delimiter, in this case `$$ $$` and `$$ $$`. The command `\( \)` also same as `$ $` and `\[ \]` is same as `$$ $$` .

Formulas are displayed in math mode, where variables are in italics and numbers appear in roman typeface. Spaces are ignored when writing the equation — LaTeX will insert the appropriate space based on the character. Equations that are in-line with text are enclosed in dollar signs (i.e. `$...$`). The formula, $a^2 + b^2 = c^2$, is entered as $a^2+b^2=c^2$. Exponents or superscripts are preceded by ^ and subscripts are preceded by _ . Many exponents, superscripts, and subscripts consist of more than one character. Be sure to include curly brackets around the exponent, superscript, or subscript in those cases (e.g. a_{11} is entered as a_{11}).

LaTeX allows two writing modes for mathematical expressions: the **inline** mode and the **display** mode. The first one is used to write formulas that are part of a text. The

second one is used to write expressions that are not part of a text or paragraph, and are therefore put on separate lines.

```
The mass-energy equivalence is described by the famous equation
$$E=mc^2$$

discovered in 1905 by Albert Einstein.  In natural units ($c$ = 1),
the formula expresses the identity

\begin{equation}
E=m
\end{equation}
```

The mass-energy equivalence is described by the famous equation

$$E = mc^2$$

discovered in 1905 by Albert Einstein. In natural units (c = 1), the formula expresses the identity

$$E = m \tag{1}$$

The displayed mode has two versions: numbered and unnumbered. To print your equations in display mode use one of these delimiters: `\[ \]`, `$$ $$`, `\begin{equation} \end{equation}` or `\begin{align} \end{align}`

To get unnumbered equation in place of `equation` and `align` we need to use `equation*` and `align*` but for this we need *amsmath* packages

4.1.1 Alignment

The *amsmath* package provides a handful of options for displaying equations. You can choose the layout that better suits your document, even if the equations are really long, or if you have to include several equations in the same line.

When you will put `equation*` no equation number will appear. Before that let us include `\usepackage{amsmath}` in the preamble of your document.

```
\begin{equation*}
e^{\pi i} + 1 = 0
\end{equation*}
```

$$e^{\pi i} + 1 = 0$$

For equations longer than a line use the `multline` environment. Insert a `\\` to set a point for the equation to be broken. The first part will be aligned to the left and the second part will be displayed in the next line and aligned to the right.

```
\begin{multline*}
\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}
-\frac{x^7}{7!}+\frac{x^9}{9!} \\
-\frac{x^{11}}{11!}+\frac{x^{13}}{13!}
-\cdots
\end{multline*}
```

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \frac{x^{13}}{13!} - \cdots$$

Sometimes an adjustment of space is needed. Use `\,` for a small space, `\:` for a medium space, `\;` for a large space, and `\!` for negative space. If there are several equations that you need to align vertically, the align environment will do it:

```
\begin{align}
        2x - 5y &=  8 \\
        3x + 9y &=  -12\\
        7x + 3y &= 6
\end{align}
```

$$2x - 5y = 8 \tag{2}$$
$$3x + 9y \leq 2$$
$$7x + 3y \geq 6 \tag{3}$$

If an equation is longer than one line or several formulas must be grouped together, the `eqnarray` and `align` environment could be used. Usually, the `eqnarray` environment is only used if authors cannot use the `amsmath` package. The special version, `eqnarray*` and `align*`, suppresses equation numbers. The `eqnarray*` environment behaves like a `tabular` environment consisting of three columns, where the column positioning is right (r), center (c), and left (l), respectively. The amperstand (i.e. `&`) is still the column delimiter and the double backslash (i.e.`\\`) is still the row delimiter. Whereas `\align*` gives us flexibility to create as many as column we want but all columns with left alignment.

```
        \begin{eqnarray*}
                y &=& x^2 + 2x + 1\\
                &=& (x+1)(x+1) \\
                &=& (x+1)^2
        \end{eqnarray*}
```

$$\begin{aligned} y &= x^2 + 2x + 1 \\ &= (x+1)(x+1) \\ &= (x+1)^2 \end{aligned}$$

As you notice `\nonumber` command use to remove numbering from that particular equation and `&` operator use to specify alignment, that is, it determines where the equations align. Let us check a more complex example with unnumbered equation format, that means, `align*`:

```
\begin{align*}
x&=y          &  w &=z              &  a&=b+c\\
2x&=-y        &  3w&=\frac{1}{2}z   &  a&=b\\
-4 + 5x&=2+y  &  w+2&=-1+w          &  ab&=cb
\end{align*}
```

$$\begin{aligned} x &= y & w &= z & a &= b+c \\ 2x &= -y & 3w &= \frac{1}{2}z & a &= b \\ -4+5x &= 2+y & w+2 &= -1+w & ab &= cb \end{aligned}$$

Here we arrange the equations in three columns. LaTeX assumes that each equation consists of two parts separated by a `&`; also that each equation is separated from the one before by an `&`.

Again, use `*` to toggle the equation numbering. When numbering is allowed, you can label each row individually.

To number many equation as a single equation number we can use `split` command.

$$\begin{aligned} A &= \frac{\pi r^2}{2} \\ &= \frac{1}{2}\pi r^2 \end{aligned} \tag{4}$$

4.2 Label and Tag

In LaTeX you can easily reference almost anything that is numbered (sections, figures, formulas), and LaTeX will take care of numbering, updating it whenever necessary. The commands to be used do not depend on what you are referencing, and they are:

- `\label{marker}`
  you give the object you want to reference a marker, you can see it like a name.
- `\ref{marker}`
  you can reference the object you have marked before. This prints the number that was assigned to the object.
- `\pageref{marker}`
  It will print the number of the page where the object is.

LaTeX will calculate the right numbering for the objects in the document; the marker you have used to label the object will not be shown anywhere in the document. Then LaTeX will replace the string "`\ref{marker}`" with the right number that was assigned to the object. If you reference a marker that does not exist, the compilation of the document will be successful but LaTeX will return a warning:

LaTeX Warning: There were undefined references.

```
\begin{align}
x^2 - 5 x + 6 = 0\label{eq:1} \\
\end{align}

Using the formula $\displaystyle x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ we have

\begin{align}
x_1 &= \frac{5 + \sqrt{25 - 4 \times 6}}{2} = 3\label{eq:2}\\
x_2 &= \frac{5 - \sqrt{25 - 4 \times 6}}{2} = 2\label{eq:3}
\end{align}

and so we have solved equation~\ref{eq:1}  and see the solution
```

```
in page \pageref{eq:3}.
```

$$x^2 - 5x + 6 = 0 \tag{5}$$
(6)

Using the formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ we have

$$x_1 = \frac{5 + \sqrt{25 - 4 \times 6}}{2} = 3 \tag{7}$$

$$x_2 = \frac{5 - \sqrt{25 - 4 \times 6}}{2} = 2 \tag{8}$$

and so we have solved equation 5 and see the solution in page 38.

As you can see, the label is placed soon after the end of that equation. In order to reference a formula, you have to use an environment that adds numbers. Most of the times you will be using the `equation` or `align` environment.

- `eqref{marker}`
  It works exactly like `\ref{}`, but it adds parentheses so that, instead of printing a plain number as 5, it will print (5). This can be useful to help the reader distinguish between formula and other things.

- `\tag{eqnno}`
  The `\tag{eqnno}` command is used to manually set equation numbers where eqnno is the arbitrary text string you want to appear in the document. This may be useful if you want to repeat an equation that is used before, e.g. `\tag{\ref{eqn:before}}`.

- `\notag`
  If you use `\notag` after a particular equation it will tell compiler there will be no equation number corresponding to that particular equation. This command work same as `\nonumber`

- `\numberwithin{countera}{counterb}`
  This command replaces the simple `countera` by a more sophisticated `counterb.countera`. For example `\numberwithin{equation}{section}` in the preamble will prepend the section number to all equation number.

The maths styles can be set explicitly. For instance, if you want an in-line mathematical element to display as a equation-like element put `\displaystyle` before that element as we have used in our earlier example. There are other type of styles are available which can be used inside math mode, for example, `\textstyle`, `\scriptstyle`, `\scriptscriptstyle`.

```
In-line maths:  $f(x) = \displaystyle
 \frac{1}{1+x}$. The same is true
  the other way around:

\begin{eqnarray*}
f(x) &=\sum_{i=0}^{n} \frac{a_i}{1+x}\\
\textstyle f(x) &= \textstyle
\sum_{i=0}^{n} \frac{a_i}{1+x} \\
\scriptstyle f(x) &= \scriptstyle
\sum_{i=0}^{n} \frac{a_i}{1+x} \\
\scriptscriptstyle f(x) &=
\scriptscriptstyle \sum_{i=0}^{n}
\frac{a_i}{1+x}
\end{eqnarray*}
```

In-line maths: $f(x) = \displaystyle \frac{1}{1+x}$. The same is true the other way around:

$$\begin{aligned}
f(x) &= \sum_{i=0}^{n} \frac{a_i}{1+x}\\
\textstyle f(x) &= \textstyle \sum_{i=0}^{n} \frac{a_i}{1+x} \\
\scriptstyle f(x) &= \scriptstyle \sum_{i=0}^{n} \frac{a_i}{1+x} \\
\scriptscriptstyle f(x) &= \scriptscriptstyle \sum_{i=0}^{n} \frac{a_i}{1+x}
\end{aligned}$$

4.3 Exponent and Brackets

Let us look into following example.

```
\[ \lim_{n\to\infty} \int_0^1 x^n + y^n \ dx \]
```

$$\lim_{n\to\infty} \int_0^1 x^n + y^n \ dx$$

In LaTeX , subscripts and superscripts are written using the symbols ^ and _, in this case the x and y exponents where written using these codes. The codes can also be used in some types of mathematical symbols, in the integral included in the example the _ is used to set the lower bound and the ^ for the upper bound.

The command \limits changes the way the limits are displayed in the integral, if not present the limits would be next to the integral symbol instead of being on top and bottom.

```
\[ \lim_{n\to\infty} \int\limits_0^1
                        x^n + y^n \ dx \]
```

$$\lim_{n\to\infty} \int\limits_0^1 x^n + y^n \ dx$$

We can subscripts and superscripts in various operators.

```
\[ \sum_{i=1}^{\infty} \frac{1}{n^s}
                = \prod_p \frac{1}{1 - p^{-s}} \]
```

$$\sum_{i=1}^{\infty} \frac{1}{n^s} = \prod_p \frac{1}{1 - p^{-s}}$$

Following are some mathematical operators.

$\int$ \int	$\iiint$ \iiint	$\cup$ \cup
$\iint$ \iint	$\oint$ \oint	$\cap$ \cap
$\amalg$ \amalg	$\vee$ \vee	$\wedge$ \wedge
$\oplus$ \oplus	$\ominus$ \ominus	$\otimes$ \otimes
$\prod$ \prod	$\cong$ \cong	$\sqcup$ \sqcup

You can use each operator given in above in big size.

$\bigvee$ \bigvee	$\bigwedge$ \bigwedge	$\bigcup$ \bigcup
$\bigoplus$ \bigoplus	$\bigotimes$ \bigotimes	$\bigcap$ \bigcap

There are various type of arrows availabe in LaTeX

$\rightarrow$	\rightarrow, \to	$\mapsto$	\mapsto
$\nrightarrow$	\nrightarrow	$\longmapsto$	\longmapsto
$\longrightarrow$	\longrightarrow	$\leftarrow$	\leftarrow
$\Rightarrow$	\Rightarrow	$\leftrightarrow$	\leftrightarrow
$\nRightarrow$	\nRightarrow	$\downarrow$	\downarrow
$\Longrightarrow$	\Longrightarrow	$\uparrow$	\uparrow
$\leadsto$	\leadsto	$\updownarrow$	\updownarrow

The right arrows in the first column have matching left arrows, such as \leftarrow for $\leftarrow$, \uparrow for $\uparrow$ and \downarrow for $\downarrow$ also \curvearrowleft for $\curvearrowleft$ etc.

Parentheses and brackets are very common in mathematical formulas. You can easily control the size and style of brackets in LaTeX . Except {} bracket other bracket can be given from keyboard itself. To get {} bracket we need to use \{ \}, respectively.

The size of the brackets can be controlled explicitly. The commands \Bigg and \bigg stablish the size of the corresponding delimiters.

```
(\big( \Big( \bigg( \Bigg(\\
]\big] \Big] \bigg] \Bigg]\\
\{ \big\{ \Big\{ \bigg\{ \Bigg\{\\
\langle \big \langle \Big \langle \bigg \langle
                \Bigg \langle\\
\rangle \big \rangle \Big \rangle \bigg \rangle
                \Bigg \rangle\\
```

$$\big(\Big(\bigg(\Bigg(\\ ]\big] \Big] \bigg] \Bigg]\\ \{ \big\{ \Big\{ \bigg\{ \Bigg\{\\ \langle \big \langle \Big \langle \bigg \langle \Bigg \langle\\ \rangle \big \rangle \Big \rangle \bigg \rangle \Bigg \rangle\\$$

The size of the brackets can be manually set, or they can be resized dynamically in your document, as shown in the next example:

```
\[
\left \{
\begin{tabular}{ccc}
1 & 5 & 8 \\
0 & 2 & 4 \\
3 & 3 & -8
\end{tabular}
\right \}
\]
```

$$\left \{ \begin{tabular}{ccc} 1 & 5 & 8 \\ 0 & 2 & 4 \\ 3 & 3 & -8 \end{tabular} \right \}$$

The \left and \right commands are used to dynamically adjust the brackets according to content. Even if you are using only one bracket both commands are mandatory.

Notice that using this command we can easily write matrix in LaTeX . For example

```
\[
        \left[
        \begin{tabular}{ccc}
        1 & 5 & 8 \\
        0 & 2 & 4 \\
        3 & 3 & -8
        \end{tabular}
        \right]
\]
```

$$\left[\begin{tabular}{ccc} 1 & 5 & 8 \\ 0 & 2 & 4 \\ 3 & 3 & -8 \end{tabular}\right]$$

The `array` environment is commonly used to write matrix and other expression in LaTeX . The `array` environment work same as `tabular` environment.

First of all, note the use of `\left` and `\right` to produce the dynamically large delimiters around the arrays. As we have already seen, if we use `\left) ... \right)`. For example

```
The \emph{characteristic polynomial} $\chi(\lambda)$ of the
$3 \times 3$~matrix
\[ \left( \begin{array}{ccc}
a & b & c \\
d & e & f \\
g & h & i \end{array} \right)\]
is given by the formula
\[ \chi(\lambda) = \left| \begin{array}{ccc}
\lambda - a & -b & -c \\
-d & \lambda - e & -f \\
-g & -h & \lambda - i \end{array} \right|.\]
```

The *characteristic polynomial* $\chi(\lambda)$ of the 3×3 matrix

$$\left(\begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array} \right)$$

is given by the formula

$$\chi(\lambda) = \left| \begin{array}{ccc} \lambda - a & -b & -c \\ -d & \lambda - e & -f \\ -g & -h & \lambda - i \end{array} \right|.$$

Now each of the c's in `{ccc}` represents a column of the matrix and indicates that the entries of the column should be centred. If the `c` were replaced by `l` then the corresponding column would be typeset with all the entries left-justified, and `r` would produce a column with all entries right-justified.

```
\[ |x| = \left\{ \begin{array}{rl}
x & \mbox{if $x \geq 0$};\\
-x & \mbox{if $x < 0$}.\end{array}
 \right. \]
```

$$|x| = \left\{ \begin{array}{rl} x & \mbox{if } x \geq 0; \\ -x & \mbox{if } x < 0. \end{array} \right.$$

Here `\mbox{text}` enable us to write text inside math mode. Note that above expression can also be written in `cases` environment. It have two column with left-justified.

```
\[ |x| = \begin{cases}
x & \mbox{if } x \geq 0;\\
-x & \mbox{if } x < 0.\end{cases}
\]
```

$$|x| = \begin{cases} x & \text{if } x \geq 0; \\ -x & \text{if } x < 0. \end{cases}$$

4.4 Matrix

Although we have learn how to write matrix in LaTeX but apart from this `amsmath` package provides several environments for matrices: `matrix`, `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix`, and `Vmatrix`. A nice feature about these commands are that the brackets are included with the environment. The `matrix` environment has none, `pmatrix` is enclosed in parenthesis, `bmatrix` is enclosed in square brackets, `Bmatrix` is enclosed in curly brackets, `vmatrix` is enclosed in vertical lines, and `Vmatrix` is enclosed in double vertical lines. An example of `bmatrix`, `vmatrix`, and `Vmatrix` is

```
\begin{align*}
\begin{bmatrix}  \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix}
&= \begin{bmatrix}
a_{11} & a_{12} & a_{13}\\
a_{21} & a_{22} & a_{23}\\
a_{31} & a_{32} & a_{33}
\end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3  \end{bmatrix} \\
\det(A) &=
\begin{vmatrix}
a_{11} & a_{12} & a_{13}\\
a_{21} & a_{22} & a_{23}\\
a_{31} & a_{32} & a_{33}
\end{vmatrix}\\
\begin{Vmatrix}
a_{11} & a_{12} & a_{13}\\
a_{21} & a_{22} & a_{23}\\
a_{31} & a_{32} & a_{33}
\end{Vmatrix}_2^2
&\le
\begin{Vmatrix}
a_{11} & a_{12} & a_{13}\\
a_{21} & a_{22} & a_{23}\\
a_{31} & a_{32} & a_{33}
\end{Vmatrix}_1
\begin{Vmatrix}
a_{11} & a_{12} & a_{13}\\
a_{21} & a_{22} & a_{23}\\
a_{31} & a_{32} & a_{33}
\end{Vmatrix}_\infty
\end{align*}
```

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$\left\| \begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \right\|_2^2 \leq \left\| \begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \right\|_1 \left\| \begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \right\|_\infty$$

4.5 Spacing

Spacing in maths mode is useful in several situations. The spacing depends on the command you insert, the example below contains a complete list of spaces and how they look like.

```
\begin{align*}
f(x) =& x^2\! +3x\! +2 \\
f(x) =& x^2+3x+2 \\
f(x) =& x^2\, +3x\, +2 \\
f(x) =& x^2\: +3x\: +2 \\
f(x) =& x^2\; +3x\; +2 \\
f(x) =& x^2\ +3x\ +2 \\
f(x) =& x^2\quad +3x\quad +2 \\
f(x) =& x^2\qquad +3x\qquad +2\\
f(x) =& \{ z\in \mathbb{C}\, |\, |z|<1\}
        \quad \textrm{and} \quad
        f(x)=\partial{S}
\end{align*}
```

$$\begin{aligned}
f(x) =& x^2+3x+2 \\
f(x) =& x^2+3x+2 \\
f(x) =& x^2\,+3x\,+2 \\
f(x) =& x^2\:+3x\:+2 \\
f(x) =& x^2\;+3x\;+2 \\
f(x) =& x^2\ +3x\ +2 \\
f(x) =& x^2\quad +3x\quad +2 \\
f(x) =& x^2\qquad +3x\qquad +2 \\
f(x) =& \{z \in \mathbb{C}\,|\,|z|<1\} \quad \text{and} \quad f(x) = \partial S
\end{aligned}$$

Description of spacing commands.

LaTeX Code	Description
\quad	space equal to the current font size (= 18 mu)
\,	3/18 of \quad (= 3 mu)
\:	4/18 of \quad (= 4 mu)
\;	5/18 of \quad (= 5 mu)
\!	-3/18 of \quad (= -3 mu)
\	equivalent of space in normal text
\qquad	twice of \quad (= 36 mu)

4.6 Chemical Equation

There is the mhchem package which we use to write chemical expression. To write chemical formulas you need to include following line in your preamable.

\usepackage{mhchem} In fact, some class files recommend it. The ce command can be used in text or math mode. Spaces are important when using the ce command which differs from most cases in LaTeX . Let's try a simple example

```
\[\ce{2H2O2(aq) ->[\text{Pt}] 2H2O(l) + O2(g)}\]
```

$$\ce{2H2O2(aq) ->[\text{Pt}] 2H2O(l) + O2(g)}$$

Complicated chemical equations can be displayed using the `ce` command.

```
\ce{2AgNO3 + BaCl2 <=>[\ce{H2O}][\ce{NH3}] 2AgCl + Ba(NO3)2}

\ce{2NH4+ + Mg -> Mg^2+ + 2NH3 + H2}

\ce{Zn^2+
        <=>[+ 2OH-][+ 2H+]
        $\underset{\text{amphoteres Hydroxid}}{\ce{Zn(OH)2 v}}$
        <=>[+ 2OH-][+ 2H+]
        $\underset{\text{Hydroxozikat}}{\ce{[Zn(OH)4]^2-}}$
}
```

$$\ce{2AgNO3 + BaCl2 <=>[\ce{H2O}][\ce{NH3}] 2AgCl + Ba(NO3)2}$$

$$\ce{2NH4+ + Mg -> Mg^2+ + 2NH3 + H2}$$

$$\ce{Zn^2+ <=>[+2OH-][+2H+] $\underset{\text{amphoteres Hydroxid}}{\ce{Zn(OH)2 v}}$ <=>[+2OH-][+2H+] $\underset{\text{Hydroxozikat}}{\ce{[Zn(OH)4]^2-}}$}$$

The double harpoon and the text surrounding it were created by inputing `<=>[text above][text below]`.

Few examples are given below.

Type	Code	Output
Formulae	`\ce{H2O}`	$\ce{H2O}$
	`\ce{Sb2O3}`	$\ce{Sb2O3}$
	`\ce{$n^2$ H2O}`	$\ce{$n^2$ H2O}$
Charges	`\ce{CrO4^2-}`	$\ce{CrO4^2-}$
	`\ce{[AgCl2]-}`	$\ce{[AgCl2]-}$
	`\ce{Y^{99+}}`	$\ce{Y^{99+}}$
Isotopes	`\ce{^{227}_{90}Th+}`	$\ce{^{227}_{90}Th+}$
	`\ce{^{0}_{-1}n^{-}}`	$\ce{^{0}_{-1}n^{-}}$
	`\ce{Y^{99+}}`	$\ce{Y^{99+}}$

5. Basic Drawing in LaTeX

One way to draw graphics directly with TeX commands is PGF/TikZ. TikZ can produce portable graphics in both PDF and PostScript formats using either plain (pdf)TEX, (pdf)Latex or ConTEXt.

PGF ("portable graphics format") is the basic layer, providing a set of basic commands for producing graphics, and TikZ ("TikZ ist kein Zeichenprogramm") is the frontend layer with a special syntax, making the use of PGF easier. TikZ commands are prevalently similar to Metafont, the option mechanism is similar to PsTricks syntax.

5.1 Introduction

Using TikZ in a LaTeX document requires loading the tikz package: `\usepackage{tikz}` somewhere in the preamble. This will automatically load the pgf package. To load further libraries use

```
\usetikzlibrary{list of libraries separated by commas}
```

Examples for libraries are “arrows", “automata", “backgrounds", “calendar", “chains", “matrix", “mindmap", “patterns", “petri", “shadows", “shapes.geometric", “shapes.misc", “spy", “trees".

Drawing commands have to be enclosed in an tikzpicture environment

```
\begin{tikzpicture}[options]
tikz commands
\end{tikzpicture}
```

In the place of `options` one can use following command:

- `baseline=dimension` Without that option the lower end of the picture is put on the baseline of the surrounding text. Using this option, you can specify that the picture should be raised or lowered such that the height dimension is on the baseline.
- `scale=factor` option to scale the entire picture.
- `xscale=2.5, yscale=0.5` scale tikz picture different for height and width.

Coordinates are specified in round brackets in an arbitrary TEX dimension either using Cartesian coordinates (comma separated), e.g. 1cm in the x direction and 2pt in the y direction, `(1cm,2pt)` or using polar coordinates (colon separated), e.g. 1cm in 30 degree direction, `(30:1cm)`.

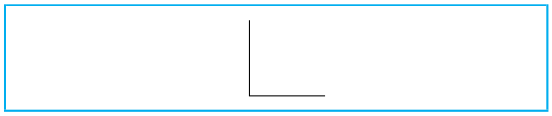
Without specifying a unit $(1, 2)$, the standard one is cm $(1cm, 2cm)$.

The following series of commands are used for various purpose in tikz which will discuss later. "`\draw`", "`\fill`", "`\filldraw`", "`\pattern`", "`\shade`", "`\shadedraw`", "`\clip`", "`\useasboundingbox`"

5.2 Basic Drawing

Straight lines are given by coordinates separated by a double minus,

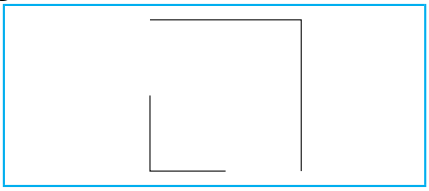
```
\begin{tikzpicture}
\draw (1,0) -- (0,0) -- (0,1);
\end{tikzpicture}
```



First, you declare a *tikzpicture* environment, before this you must include the line `\usepackage{tikz}` in the preamble of your document. The first coordinate represents a move-to operation. This is followed by a series of "path extension operations", like "–(coordinates)".

The above picture can also be drawn using following command.

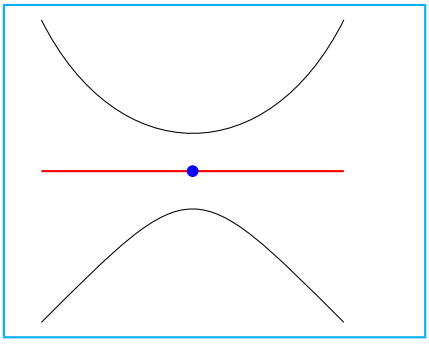
```
                \begin{tikzpicture}
\draw (1,0) -| (0,1);
\draw (2,0) |- (0,2);
                \end{tikzpicture}
```



Two points can be connected by straight lines that are only horizontal and vertical.

- `(p1) -| (p2)` connect (p1) and (p2) using horizontal and vertical line travel **first horizontal and then vertical**.
- `(p1) |- (p2)` connect (p1) and (p2) using horizontal and vertical line travel **first vertical and then horizontal**.

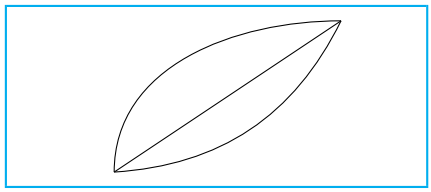
```
\begin{tikzpicture}
\draw (-2,0) -- (2,0);
\filldraw [gray] (0,0) circle (2pt);
\draw (-2,-2) .. controls (0,0) .. (2,-2);
\draw (-2,2) .. controls (-1,0)
                and (1,0) .. (2,2);
\end{tikzpicture}
```



- `\draw[red,thick] (-2,0, 0) -- (2,0);` This defines a **line** whose endpoint are $(-2, 0)$ and $(2, 0)$ whose colour is red and with a thick stroke.
- `\filldraw [blue] (0,0) circle (2pt);` The point is created as a very small blue circle centred at $(0, 0)$ and whose radius is $(2pt)$. The `\filldraw` command is used in to draw elements and fill them with some specific colour.
- `\draw (-2,2) .. controls (-1,0) and (1,0) .. (2,2);` Draws a curve, is a bit tricky at first. There are 4 points defining it: $(-2, 2)$ and $(2, 2)$ are its endpoints, $(-1, 0)$ and $(1, 0)$ are control points (can be equal) that determine 'how curved' it is. You can think of these two points as "attractor points".

User-defined paths can be created using the "`to`" operation. Without an option it corresponds to a straight line, exactly like the double minus command. Using the "out" and "in" option a curved path can created. E.g. "`[out=135,in=45]`" causes the path to leave at an angle of 135 degree at the first coordinate and arrive at an angle of 45 degree at the second coordinate.

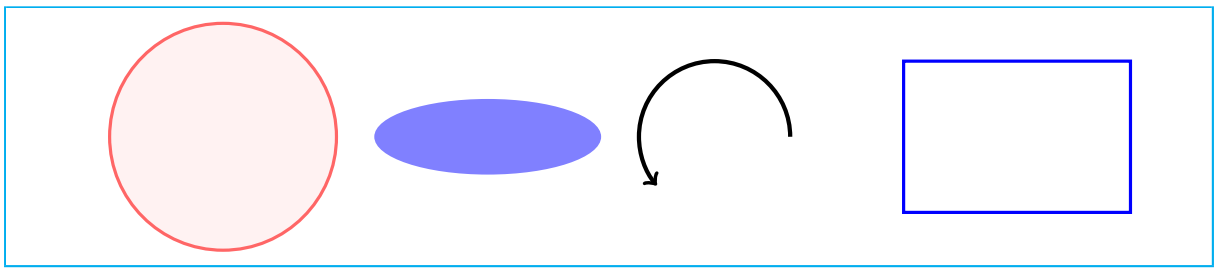
```
\begin{tikzpicture}
\draw (0,0) to (3,2);
\draw (0,0) to[out=90,in=180] (3,2);
\draw (0,0) to[bend right] (3,2);
\end{tikzpicture}
```



5.3 Geometric Figure

Geometric figures can be made up from simpler elements or created by an special command. Let's start with circles, ellipses and arcs.

```
\begin{tikzpicture}
\filldraw[color=red!60, fill=red!5, very thick](-1,0) circle (1.5);
\fill[blue!50] (2.5,0) ellipse (1.5 and 0.5);
\draw[ultra thick, ->] (6.5,0) arc (0:220:1);
\draw[blue, very thick] (8,-1)rectangle (11,1);
\end{tikzpicture}
```



- `\filldraw[color=red!60, fill=red!5, very thick](-1,0) circle (1.5);` We are already familiar with this command, it was used to draw a point in the previous section, but here there are some additional parameters inside brackets. These are explained below.

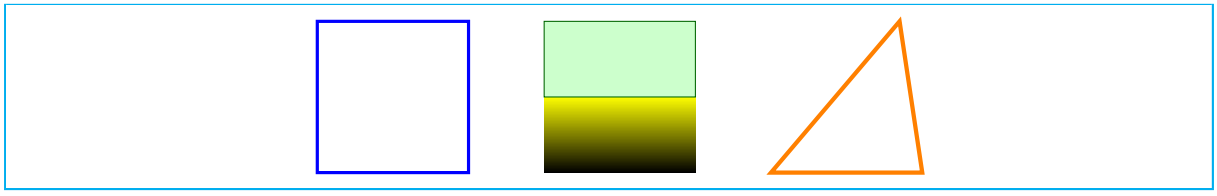
  - `color=red!60` The colour of the ring around the circle is set to 60% red and 40% white.
  - `fill=red!5` The circle is filled in with an even more lighter red.
  - `very thick` This parameter defines the thickness of the stroke, can also be used in the `\fill` command.

- `\fill[blue!50] (2.5,0) ellipse (1.5 and 0.5);` To create an ellipse you provide a centre point (2.5,0), and two radii: horizontal and vertical (1.5 and 0.5). Also notice the command fill instead of draw or filldraw, this is because in this case there's no need to control outer and inner colours.

- `\draw[ultra thick, ->] (6.5,0) arc (0:220:1);` This command will draw an arc, the extra parameter `->` indicates that the arc will have an arrow at the end. To draw the arc you must provide the centre point and the other three numbers: the starting and ending angles, and the radius; in the format `(0:220:1)`.

- `\draw[blue, very thick] (0,0)rectangle (3,2);` **Rectangles** are created by the special command rectangle. You have to provide two points, the first one is where the "pencil" begins the draw and the second one is the diagonally opposite corner point.

```
\begin{tikzpicture}
\draw[blue, very thick] (-3,0)rectangle (-1,2);
\shade[top color=yellow, bottom color=black] (0,1) rectangle (2,0);
\filldraw[fill=green!20!white, draw=green!40!black] (0,1) rectangle (2,2);
\draw[orange, ultra thick] (3,0) -- (5,0) -- (4.7,2) -- cycle;
\end{tikzpicture}
```



- `\draw[orange, ultra thick] (3,0) -- (5,0) -- (4.7,2) -- cycle;` To draw a polygon we draw a closed path of straight lines: a line from (3,0) to (5,0) and a line from (5,0) to (4.7,2). Finally, the last point declared is not such thing, but the word `cycle`, meaning the next line should join the last and the first points.

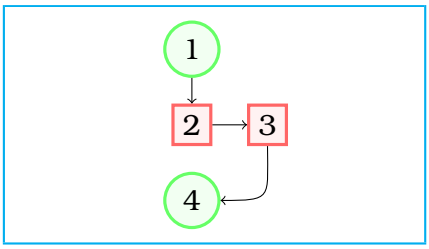
- `\shade[top color=yellow, bottom color=black] (0,1) rectangle (2,0);` It will create rectangle filled with a mixture color gradient, where top color is yellow and bottom color is black.

- `\filldraw[fill=green!20!white, draw=green!40!black] (0,1) rectangle (2,2);` It will also create rectangle filled with a color mixed with 40% green and 60% black, and its border line is of color black.

The nodes are probably the most versatile elements in **Tikz**, We already used one node in the introduction to add some text to the figure. In the next example nodes will be

used to create a diagram.

```
\begin{tikzpicture}[
roundnode/.style={circle, draw=green!60,
         fill=green!5, very thick,
         minimum size=7mm},
squarednode/.style={rectangle, draw=red!60,
        fill=red!5, very thick,
        minimum size=5mm},
]
%Nodes
\node[squarednode]      (maintopic) {2};
\node[roundnode]        (uppercircle)
                [above=of maintopic] {1};
\node[squarednode]      (rightsquare)
                [right=of maintopic] {3};
\node[roundnode]        (lowercircle)
                [below=of maintopic] {4};

%Lines
\draw[->] (uppercircle.south) --
                (maintopic.north);
\draw[->] (maintopic.east) --
                        (rightsquare.west);
\draw[->] (rightsquare.south) ..
                controls +(down:7mm)
                and +(right:7mm) ..
                (lowercircle.east);
\end{tikzpicture}
```

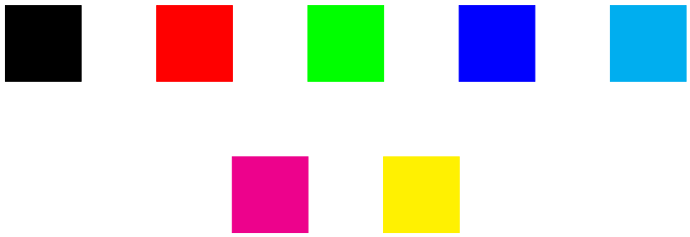
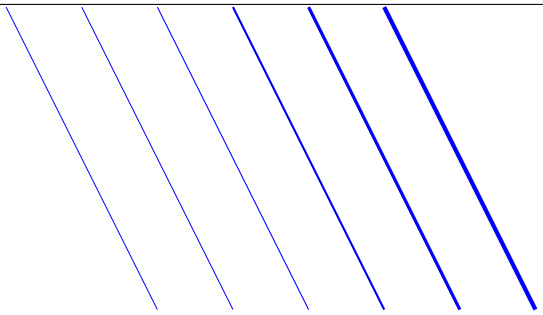


There are essentially three commands in this figure: A node definition, a node declaration and lines that join two nodes.

- `roundnode/.style={circle, draw=green!60, fill=green!5, very thick, minimum size=7mm}` Passed as a parameter to the tikzpicture environment defines a node that will be referred as `roundnode`, this node will be a circle whose outer ring will be `gren!60` and will be coloured with `green!5`, the stroke will be `very thick` and its minimum size is 7mm. The line below this defines a second rectangle-shaped node called `squarednode` with similar parameters.
- `\node[squarednode] (maintopic) {2};` This will create a `squarednode`, as defined in the previous command. This node will have an id, `maintopic` and will contain the number 2, if you leave an empty space inside the braces no text will be displayed.
- `[above=of maintopic]` Notice that all but the first node have an additional parameter, this parameter determines its position relative to other node. For instance `[above=of maintopic]` means that this node should appear above the node named `maintopic`. For this positioning system to work you have to add `\usetikzlibrary{positioning}` to your preamble.
- `\draw[->] (uppercircle.south) -- (maintopic.north);` An arrow-like straight line

will be drawn. We write the endpoints of the line, by referencing a node and a position relative to the node.

Possible color and thickness parameters in the tikz package:

Parameter	Values	Picture
color	white, black, red, green, blue, cyan, magenta, yellow	
thickness	ultra thin, very thin, thin, thick, very thick, ultra thick	

5.4 pgfplot Package

The pgfplots package is a powerful tool, based on tikz, dedicated to create scientific graphs. It is a visualization tool to make simpler the inclusion of plots in your documents. The basic idea is that you provide the input data/formula and pgfplots does the rest.

To include pgfplots in your document is very easy, add the next line to your preamble and that's it:

```
\usepackage{pgfplots}
```

Some additional tweaking for this package can be made in the preamble. To change the size of each plot and also guarantee compatibility backwards (recommended) add the next line:

```
\pgfplotsset{width=10cm,compat=1.9}
```

This changes the size of each pgfplot figure to 10 cementers, which is huge; you may use different units (pt, mm, in). The compat parameter is for the code to work on the package version 1.9 or latter.

The general format for drawing a figure is:

For single curve we use

```
\begin{tikzpicture}
        \begin{axis}[axis-options]
                \addplot formula;
        \end{axis}
\end{tikzpicture}
```

For multiple curve we use

```
\begin{tikzpicture}
        \begin{axis}[axis-options]
                \addplot[plot-options] formula;
                        \addlegendentry{label}
                \addplot[plot-options] formula;
                        \addlegendentry{label}
                ...
        \end{axis}
\end{tikzpicture}
```

Since pgfplot is based on tikz the plot must be inside a tikzpicture environment. Then the environment declaration \begin{axis}, \end{axis} will set the right scaling for the plot.

To add an actual plot, the command \addplot[color=red]{f(x)}; is used. Inside the squared brackets some options can be passed, in this case we set the colour of the plot to red; the squared brackets are mandatory, if no options are passed leave a blank space between them. Inside the curly brackets you put the function to plot. Is important to remember that this command must end with a semicolon ;.

5.4.1 Graph Formulas

You can personalize your plots to look exactly what you want depending on the data available.

- **function plot** To plot a curve where the vertical coordinate is a function of the horizontal coordinate, just give the function formula in terms of x within braces. For example: \addplot {-4+x^2-x^4/32};

  Trigonometric functions assume degrees, so invoke as sin(deg(x)) for example, and convert arc-functions as in atan(x)/deg(1).

- **parametric plot** parametric plot For a parametric plot give the horizontal and vertical formulas in terms of 'x' within braces, comma separated, within parentheses. For example, to plot $y = \sqrt{2x-6}$ one could do \addplot ({x^2/2+2},{x});

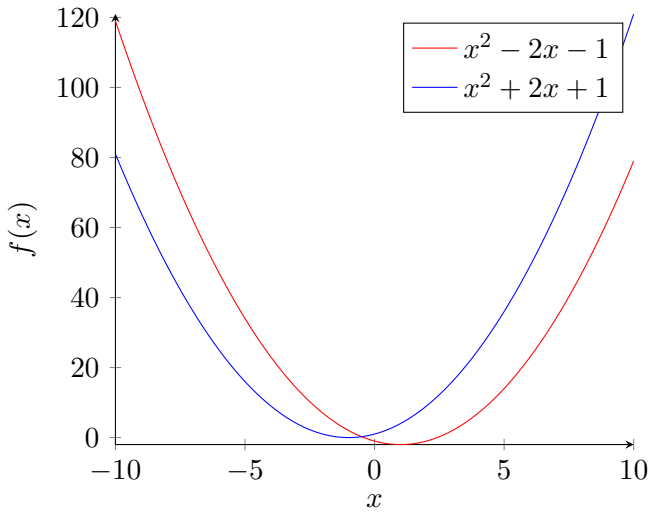
- **given data** In place of formula, use coordinatespoint-list where the point-list has the form $(x_1, y_1)(x_2, y_2)\dots(x_n, y_n)$ for the numerical data point coordinates (no commas between the parentheses). For example, to draw the absolute value function one could

  \addplot coordinates{(-2,2)(0,0)(2,2)}

Let us see the following example.

```
\begin{tikzpicture}
\begin{axis}[ axis lines = left, xlabel = $x$, ylabel = {$f(x)$},]

%Below the red parabola is defined
\addplot [
                        domain=-10:10,
                        samples=100,
                        color=red,
                ]
                {x^2 - 2*x - 1};
\addlegendentry{$x^2 - 2x - 1$}
%Here the blue parabloa is defined
```

```
        \addplot [
                                domain=-10:10,
                                samples=100,
                                color=blue,
                        ]
                        {x^2 + 2*x + 1};
        \addlegendentry{$x^2 + 2x + 1$}

        \end{axis}
        \end{tikzpicture}
```

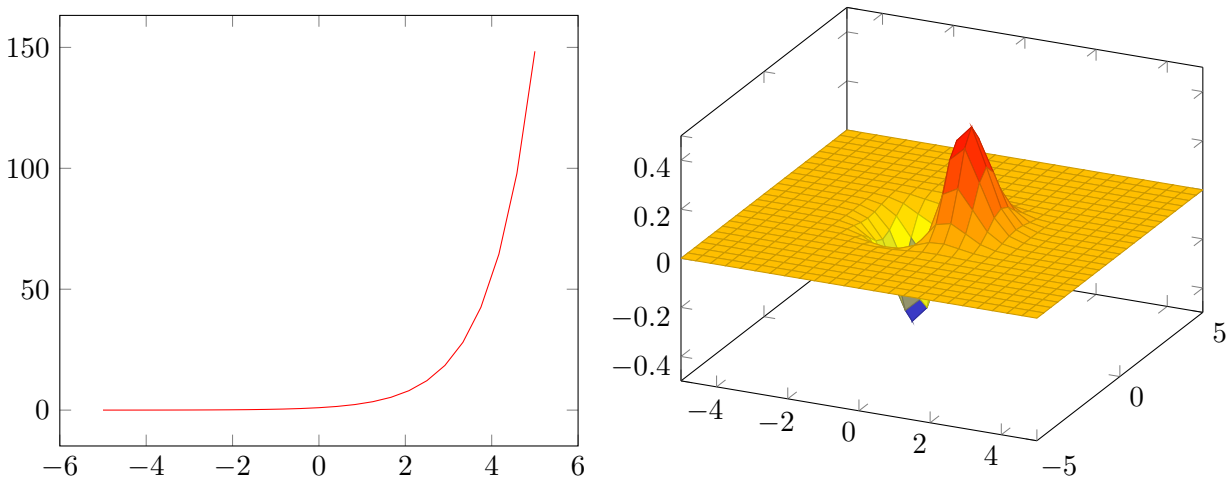


- `axis lines = left` This will set the axis only on the left and bottom sides of the plot, instead of the default box. Further customisation options at the reference guide.
- `xlabel = $x$ and ylabel = {$f(x)$}` Self-explanatory parameter names, these will let you put a label on the horizontal and vertical axis. Notice the ylabel value in between curly brackets, this brackets tell pgfplots how to group the text. The xlabel could have had brackets too. This is useful for complicated labels that may confuse pgfplot.
- `\addplot` This will add a plot to the axis, general usage was described at the introduction. There are two new parameters in this example.
- `domain=-10:10` This establishes the range of values of x.
- `samples=100` Determines the number of points in the interval defined by domain. The greater the value of samples the sharper the graph you get, but it will take longer to render.
- `\addlegendentry{$x^2 - 2x - 1$}` This adds the legend to identify the function $x^2 - 2x - 1$.

Let us look into another example.

`\addplot3[surf]]{exp(-x^2-y^2)*x};`. This will add a `3dplot`, and the option `surf` inside squared brackets declares that it's a surface plot. The function to plot must be placed inside curly brackets.

```
\begin{tikzpicture}
\begin{axis}

        \addplot[color=red]{exp(x)};

\end{axis}
\end{tikzpicture}    % Here ends the first plot
\hskip 5pt    % Here begins the 3d plot
\begin{tikzpicture}
\begin{axis}

\addplot3[surf]{exp(-x^2-y^2)*x};

\end{axis}
\end{tikzpicture}
```



To put a second plot next to the first one declare a new tikzpicture environment. Do not insert a new line, but a small blank gap, or you can use `hskip` 10pt will insert a 10pt-wide blank space.

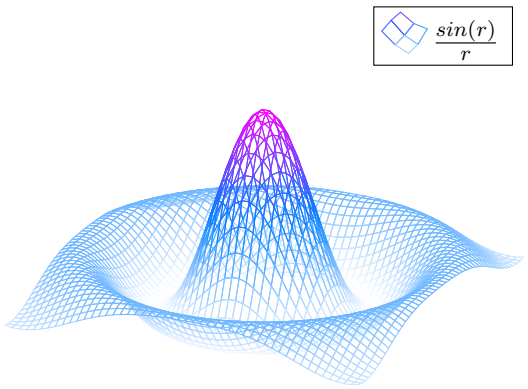
5.4.2 3D Plot

As we have seen earlier example `pgfplots` has the 3d Plotting capabilities that you may expect in a plotting software.

```
\begin{tikzpicture}
\begin{axis}[
title=Exmple using the mesh parameter,
hide axis,
colormap/cool,
]
\addplot3[
mesh,
samples=50,
```

```
        domain=-8:8,
        ]
        {sin(deg(sqrt(x^2+y^2)))/sqrt(x^2+y^2)};
        \addlegendentry{$\frac{sin(r)}{r}$}
        \end{axis}
        \end{tikzpicture}
```

Exmple using the mesh parameter

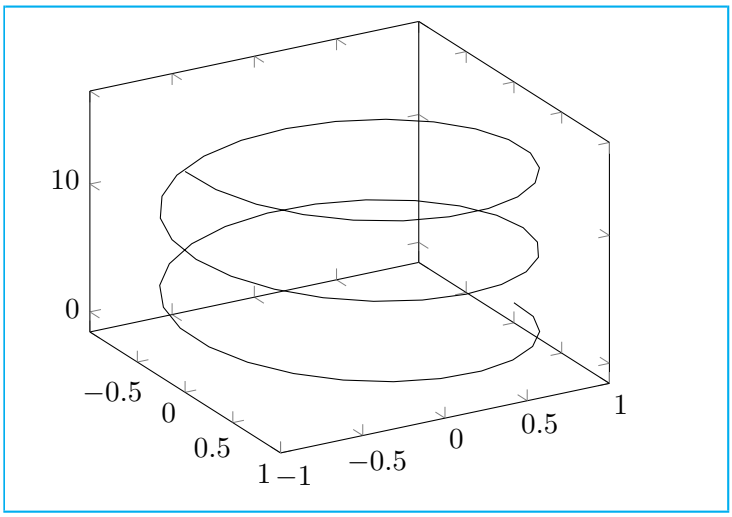


Here `colormap/cool` is the colour scheme to be used in the plot.

5.4.3 parametric Plot

The syntax for parametric plots is slightly different.

```
\begin{tikzpicture}
        \begin{axis}
        [view={60}{30}]

\addplot3[
        domain=0:5*pi,
        samples = 60,
        samples y=0,
        ]
({sin(deg(x))},
{cos(deg(x))}, {x});
\end{axis}
\end{tikzpicture}
```



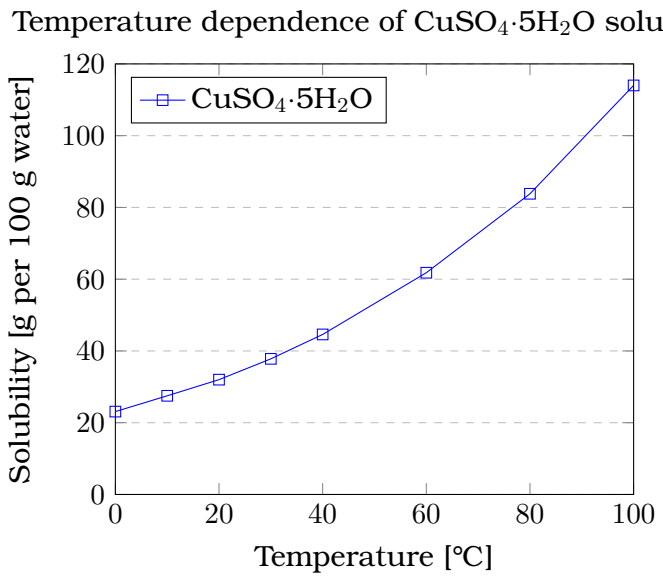
Here `view={60}{30}` This changes the view of the plot. The parameter is passed to the axis environment, which means this can be used in any other type of 3d plot. The first value is a rotation, in degrees, around the z-axis; the second value is to rotate the view around the x-axis. In this example when we combine a 60 degree rotation around the z-axis and a 30 degree rotation around the x-axis we end up with a view of the plot from side.

Also the samples y=0 to prevent pgfplots from joining the extreme points of the spiral. Note that the way the function to plot is passed to the addplot3 environment. Each parameter function is grouped inside curly brackets and the three parameters are delimited with parenthesis.

5.4.4 Plotting From data

Scientific research often yields data that has to be analysed. The next example shows how to plot data with *pgfplots*:

```
\begin{tikzpicture}
\begin{axis}[
title={Temperature dependence of CuSO$_4\cdot$5H$_2$O solubility},
xlabel={Temperature [\textcelsius]},
ylabel={Solubility [g per 100 g water]},
xmin=0, xmax=100,
ymin=0, ymax=120,
xtick={0,20,40,60,80,100},
ytick={0,20,40,60,80,100,120},
legend pos=north west,
ymajorgrids=true,
grid style=dashed,
]

\addplot[
color=blue,
mark=square,
]
coordinates {
        (0,23.1)(10,27.5)(20,32)(30,37.8)(40,44.6)(60,61.8)(80,83.8)(100,114)
};
\legend{CuSO$_4\cdot$5H$_2$O}

\end{axis}
\end{tikzpicture}
```



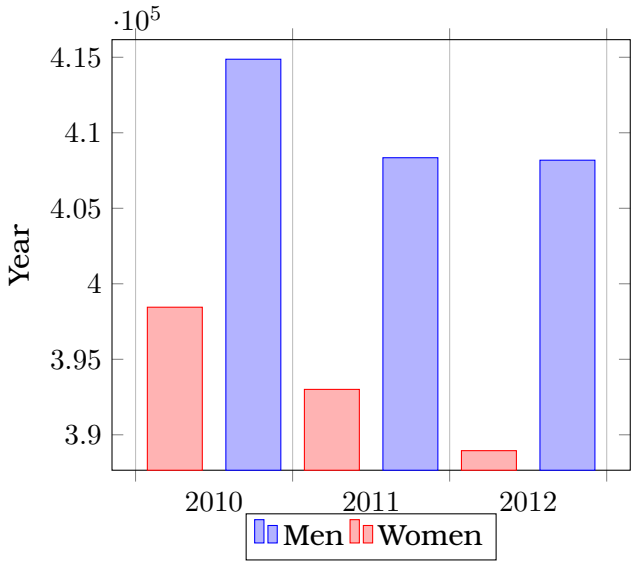
- `title={Temperature dependence of CuSO$_4\cdot$5H$_2$O solubility}` As you might expect, assigns a title to the figure. The title will be displayed above the plot.
- `xmin=0, xmax=100, ymin=0, ymax=120.` Minimum and maximum bounds of the x and y axes.
- `xtick={0,20,40,60,80,100}, ytick={0,20,40,60,80,100,120}` Points where the marks are placed. If empty the ticks are set automatically.
- `legend pos=north west` Position of the legend box. Check the reference guide for more options.
- `ymajorgrids=true` This Enables/disables grid lines at the tick positions on the y axis. Use xmajorgrids to enable grid lines on the x axis.
- `grid style=dashed.` Self-explanatory. To display dashed grid lines.
- `mark=square` This draws a squared mark at each point in the cordinates array. Each mark will be connected with the next one by a straight line.
- `coordinates {(0,23.1)(10,27.5)(20,32)...}` Coordinates of the points to be plotted. This is the data you want analyse graphically.

5.4.5 Bar Graphs

Bar graphs (also known as bar charts and bar plots) are used to display gathered data, mainly statistical data about a population of some sort. Bar plots in pgfplots are highly customisable, but here we are going to show an example that 'just works':

```
\begin{tikzpicture}
\begin{axis}[
x tick label style={
        /pgf/number format/1000 sep=},
ylabel=Year,
enlargelimits=0.05,
legend style={at={(0.5,-0.1)},
```

```
        anchor=north,legend columns=-1},
ybar interval=0.7,
]
\addplot
coordinates {(2012,408184) (2011,408348)
        (2010,414870) (2009,412156)};
\addplot
coordinates {(2012,388950) (2011,393007)
        (2010,398449) (2009,395972)};
\legend{Men,Women}
\end{axis}
\end{tikzpicture}
```



6. Flowchart and Algorithm

6.1 Algorithm

To create algorithms in LaTeX you can use `algorithm2e`, `algorithmic` or `Listings` environment. The `algorithm2e` package (first released 1995, latest updated July 2017 according to the v5.0 manual) allows typesetting algorithms with a lot of customization. Unlike `algorithmic`, `algorithm2e` provides a relatively huge number of customization options to the algorithm suiting to the needs of various use. Here we only discuss about `algotithm2e` package.

6.1.1 `algorithm2e` Package

To enable algorithm environment is LaTeX insert `\usepackage{algorithm2e}` in your preamble of the document which will load `algorithm2e` package.

Typically, the usage between `\begin{algorithm}` and `\end{algorithm}` would be

1. Declaring a set of keywords(to typeset as functions/operators), layout controls, caption, title, header text (which appears before the algorithm's main steps e.g.: Input,Output)
2. Writing the main steps of the algorithm, with each step ending with a `;`

```
\begin{algorithm}[H]
        \KwData{this text}
        \KwResult{how to write algorithm with \LaTeX2e }
        initialization\;
        \While{not at end of this document}{
                read current\;
                \eIf{understand}{
                        go to next section\;
                        current section becomes this one\;
                }{
                        go back to the beginning of current section\;
                }
        }
        \caption{How to write algorithms}
```

```
\end{algorithm}
```

Algorithm 1: How to write algorithms

Data: this text
Result: how to write algorithm with LaTeX2e

```
1 initialization;
2 while not at end of this document do
3     read current;
4     if understand then
5         go to next section;
6         current section becomes this one;
7     else
8         go back to the beginning of current section;
```

Notice that here we have used `\usepackage[ruled,vlined,linesnumbered]{algorithm2e}` in our preamble of the document instead of only `\usepackage{algorithm2e}` which gives us more nice formatting of algorithm.

- `ruled`
  It will make caption as a header name of your algorithm and will display it in the beginning of algorithm with two vertical lines as in above.
- `vlined`
  It will draw vertical line from starting of a state to end of that state.
- `linesnumbered`
  The command itself suggest that it will display algorithm with line number.

Note that every line of algorithm must have to end with `\;` except keyword variable. Let us analysis the inside code.

- `\KwData{this text}`
  This is predefined keyword variable in algorithm. It will show "this text" as data in algorithm.
- `\KwResult{how to write algorithm with \LaTeX2e }` This is also predefined keyword variable in algorithm. It will show "how to write algorithm with LaTeX2e" as result in algorithm.
- `\While{<need to satisy>}{ <code> }`
  The while command takes two arguments. First one represent the code which need to satisy by while loop and secnd argument takes execution code for whole loop. Similarly other codes will work. For example `\If{<need to satisy>}{ <code> }`, `\For{<need to satisy>}{ <code> }`.
- `\eIf{<need to satisy for if>}{ <if-code> } {<else-code>}`

You can define your own keyword text using `\SetKwInOut` command. Also you can make algorithm such that none of the line will contain semicolon using `\DontPrintSemicolon`

```
\begin{algorithm}[H]
        \DontPrintSemicolon
        \caption{\textsc{AlgoName}}\label{algo:2}

        \SetKwInOut{input}{Input}  \SetKwInOut{output}{Output}

        \input{Input of Algo}
        \output{Output of Algo}
        \eIf{condition}        {condition is true}
        {
                condition is false \tcp*{comment here}
        }
        \While{this is true}{
                something for a while \;
                something for a while \;
        }
        \For{$i$ in 1:100} {something 100 times}
\end{algorithm}
```

Algorithm 2: AlgoName

Input : Input of Algo
Output Output of Algo
:

```
1 if condition then
2 |  condition is true
3 else
4 |  condition is false                                  // comment here
5 while this is true do
6 |  something for a while
7 |  something for a while
8 for i in 1:100 do
9 |  something 100 times
```

Here `\SetKwInOut{input}{Input}` define new keyword text which will use to print input.

6.1.2 `algorithmicx` Package

The `algorithmic` package provides a number of popular constructs for algorithm designs. To use the algorithmic environment to write algorithm pseudocode we use `\begin{algorithmic}` `\end{algorithmic}`.

The command `\begin{algorithmic}` can be given the optional argument of a positive integer, which if given will cause line numbering to occur at multiples of that integer. E.g. `\begin{algorithmic}[5]` will enter the algorithmic environment and number every fifth line.

Basic commands have the following syntax:

Statement (`\State` causes a new line, can also be used in front of other commands)

```
        \State $x\gets <value>$
```

Three forms of if-statements:

```
\If{<condition>} <text> \EndIf
\If{<condition>} <text> \Else <text> \EndIf
\If{<condition>} <text> \ElsIf{<condition>} <text> \Else <text> \EndIf
```

The third form accepts as many `\ElsIf{}` clauses as required. Note that it is `\ElsIf` and not `\ElseIf`.

Loops:

```
\For{<condition>} <text> \EndFor
\ForAll{<condition>} <text> \EndFor
\While{<condition>} <text> \EndWhile
\Repeat <text> \Until{<condition>}
\Loop <text> \EndLoop
```

```
\begin{algorithmic}
        \STATE $i\gets 10$
        \IF {$i\geq 5$}
        \STATE $i\gets i-1$
        \ELSE
        \IF {$i\leq 3$}
        \STATE $i\gets i+2$
        \ENDIF
        \ENDIF
\end{algorithmic}
```

$i \leftarrow 10$
if $i \geq 5$ **then**
  $i \leftarrow i - 1$
else
  if $i \leq 3$ **then**
    $i \leftarrow i + 2$
  end if
end if

Another simple example for better understanding.

```
\begin{algorithm} % enter the algorithm environment
        \caption{Calculate $y = x^n$} % give the algorithm a caption
        \label{alg1} % and a label for \ref{} commands later in the document
        \begin{algorithmic} % enter the algorithmic environment
                \REQUIRE $n \geq 0 \vee x \neq 0$
                \ENSURE $y = x^n$
                \STATE $y \Leftarrow 1$
                \IF{$n < 0$}
                \STATE $X \Leftarrow 1 / x$
                \STATE $N \Leftarrow -n$
                \ELSE
                \STATE $X \Leftarrow x$
                \STATE $N \Leftarrow n$
                \ENDIF
                \WHILE{$N \neq 0$}
                \IF{$N$ is even}
                \STATE $X \Leftarrow X \times X$
                \STATE $N \Leftarrow N / 2$
                \ELSE[$N$ is odd]
                \STATE $y \Leftarrow y \times X$
                \STATE $N \Leftarrow N - 1$
                \ENDIF
```

```
                \ENDWHILE
        \end{algorithmic}
\end{algorithm}
```

Algorithm 3: Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \Leftarrow 1$
  if $n < 0$ **then**
    $X \Leftarrow 1/x$
    $N \Leftarrow -n$
  else
    $X \Leftarrow x$
    $N \Leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \Leftarrow X \times X$
      $N \Leftarrow N/2$
    else {N is odd}
      $y \Leftarrow y \times X$
      $N \Leftarrow N - 1$
    end if
  end while

```
\begin{algorithm}
        \caption{Euclids algorithm}\label{alg:euclid}
        \begin{algorithmic}[1]
                \Procedure{Euclid}{$a,b$}\Comment{The g.c.d. of a and b}
                \State $r\gets a\bmod b$
                \While{$r\not=0$}\Comment{We have the answer if r is 0}
                \State $a\gets b$
                \State $b\gets r$
                \State $r\gets a\bmod b$
                \EndWhile\label{euclidendwhile}
                \State \textbf{return} $b$\Comment{The gcd is b}
                \EndProcedure
        \end{algorithmic}
\end{algorithm}
```

For more details please visit https://en.wikibooks.org/wiki/LaTeX/Algorithms.

6.1.3 Listings package

`listings` package provide us to import our whole code file into LaTeXTo do add `\usepackage{listings}` in the preamble.

There are two way to tell compiler which tpe of file you are goint to include. For example after `\begin{document}` put `\lstset{language=Pascal}` to set your language pascal. Or You can specify the language while including the file with the following command: `\lstinputlisting[language=C]{source.c}`

Almost every programming languages supported by listings package. Few of them are Java, Prolog, Python, Assembler, Matlab, Ruby, C++, SQL etc.

Just a simple command will import whole code. For example

```
\lstinputlisting[language=Python]{test.py}
```

```
 import numpy as np
import scipy as sp

# Math Functions not in Numpy/Scipy
def null(A, eps=1e-15):
    """Computes the null space of the real matrix A"""
    n, m = sp.shape(A)
    if n > m :
        return sp.transpose(null(sp.transpose(A), eps))
        return null(sp.transpose(A), eps)
    u, s, vh = sp.linalg.svd(A)
    s=sp.append(s,sp.zeros(m))[0:m]
    null_mask = (s <= eps)
    null_space = sp.compress(null_mask, vh, axis=0)
    return sp.transpose(null_space)
```

As you notice that although it is include the code but its looks not so good. To make it beautiful we use \lstset command.

```
\definecolor{mygreen}{rgb}{0,0.6,0}     % Define new color
\definecolor{mygray}{rgb}{0.5,0.5,0.5}  % Define new color
\definecolor{mymauve}{rgb}{0.58,0,0.82} % Define new color


\lstset{ %
        backgroundcolor=\color{white},
        basicstyle=\footnotesize,
        breakatwhitespace=false,
        breaklines=true,
        captionpos=b,
        commentstyle=\color{mygreen},
        deletekeywords={...},
        escapeinside={\%*}{*)},
        extendedchars=true,
        frame=single,
        keywordstyle=\color{blue},
        language=Python,
        morekeywords={*,...},
        numbers=left,
        numbersep=5pt,
        numberstyle=\tiny\color{mygray},
        rulecolor=\color{black},
        showspaces=false,
        showstringspaces=false,
        showtabs=false,
```

```
        stepnumber=1,
        stringstyle=\color{mymauve},
        tabsize=2,
        title=\lstname

}

        \lstinputlisting[language=Python]{test.py}
```

```python
import numpy as np
import scipy as sp

# Math Functions not in Numpy/Scipy
def null(A, eps=1e-15):
    """Computes the null space of the real matrix A"""
    n, m = sp.shape(A)
    if n > m :
        return sp.transpose(null(sp.transpose(A), eps))
        return null(sp.transpose(A), eps)
    u, s, vh = sp.linalg.svd(A)
    s=sp.append(s,sp.zeros(m))[0:m]
    null_mask = (s <= eps)
    null_space = sp.compress(null_mask, vh, axis=0)
    return sp.transpose(null_space)
```

test.py

Explanation of above code.

- `backgroundcolor=\color{white}`
  It makes the background color is white. For this `\usepackage{color}` or `\usepackage{xcolor}` packages are Required.
- `basicstyle=\footnotesize`
  It set the size of the fonts that are used for the code.
- `breakatwhitespace=false`
  It sets automatic breaks should only happen at whitespace
- `breaklines=true`
  It enable automatic line breaking.
- `captionpos=b`
  Sets the caption-position to bottom ; possible values are (b, t, h).
- `commentstyle=\color{mygreen}`
  Color for comment.
- `deletekeywords={...}`
  If you want to delete keywords from the given language
- `escapeinside={\%*}{*)}`
  If you want to add LaTeX within your code
- `extendedchars=true`
  If you use non-ASCII characters; for 8-bits encodings only, which does not work with UTF-8 then you need to enable this.

- `frame=single`
  It draw a frame around the code; possible values are (none, single, double).
- `keywordstyle=\color{blue}`
  Default keyword color style.
- `language=Python`
  The language of the code which you want to include
- `morekeywords={*,...}`
  If you want to add more keywords to the set.
- `numbers=left`
  It tells compiler here to put the line-numbers; possible values are (none, left, right).
- `numbersep=5pt`
  It define how far the line-numbers are from the code.
- `numberstyle=\tiny\color{mygray}`
  It is used to define color style for the line-numbers.
- `rulecolor=\color{black}`
  If not set, the frame-color may be changed on line-breaks within not-black text (e.g. comments (green here))
- `showspaces=false`
  It will disable to show spaces everywhere adding particular underscores; it overrides 'showstringspaces'
- `showstringspaces=false`
  It will disable to show underline spaces within two strings.
- `showtabs=false`
  It will disable to show tabs within strings adding particular underscores.
- `stepnumber=1`
  The step between two line-numbers. If it's 1, each line will be numbered.
- `stringstyle=\color{mymauve}`
  string literal style
- `tabsize=2`
  It sets default tabsize to 2 spaces.
- `title=\lstname`
  This is used to show the filename of files included with `\lstinputlisting` in the end. Also one can use `caption` instead of `title`.

A simpler version of above code is given below.

```
\lstset{numbers=left, numberstyle=\tiny, stepnumber=1, numbersep=5pt}

        \lstinputlisting[language=c]{ccode.c}
```

```c
#include <stdio.h>

int main()
{
  int n, sum = 0, c, value;

  printf("How many numbers you want to add?\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 1; c <= n; c++)
  {
    scanf("%d", &value);
    sum = sum + value;
  }

  printf("Sum of the integers = %d\n", sum);

  return 0;
}
```

ccode.c

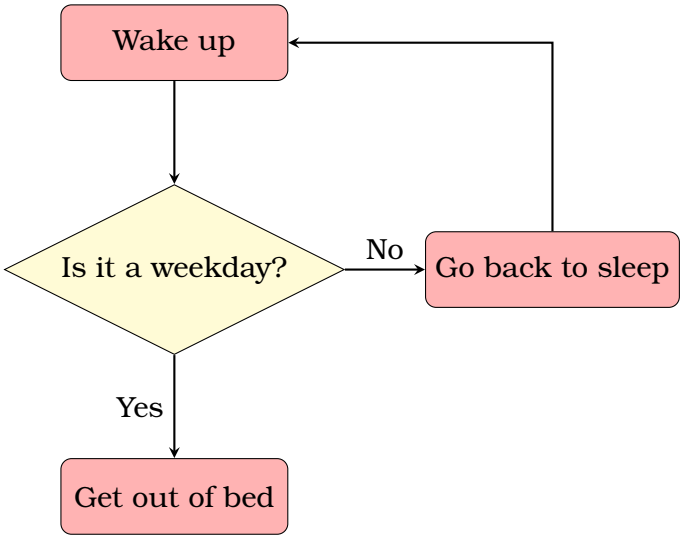
When you use figures or tables, you can add a list of them close to the table of contents; the `algorithm` package provides a similar command. Just put `\listofalgorithms`.

6.2 Flowchart

In this section were going to be looking at creating flowcharts in TikZ. To draw flowchart we need to load up the TikZ package with following library `shapes.geometric, arrows`.

```
\usepackage{tikz}
\usetikzlibrary{shapes.geometric, arrows}
```

Now we are ready to start building our flow chart. To do the we use the `tikzpicture` environment inside our document. Well create our flowchart blocks using nodes and the tikzstyles.



6.2.1 Tikzstyle Command

\tikzstyle command gives us freedom to define the block were going to use for start and stop blocks. Let us look into following example.
Here we use two type of block namely process and decision.

```
\tikzstyle{process} = [rectangle, rounded corners, minimum width=3cm,
minimum height=1cm,text centered, draw=black, fill=red!30]
```

For this block we specify a rectangle with rounded corners. We have given it a minimum width of 3cm and a minimum height of 1cm. We also ensure the text gets centred and we set both a draw and a fill colour. In this example weve set the fill colour to a colour that is 30% red mixed with 70% white.
Next we define our decision block. For this block we specify a diamond with a minimum width of 4cm and aspect ratio is 2, that means width of diamond will be twice time of its height. We also specify its fill color to 20% yellow mixed with 70% white

```
\tikzstyle{decision}=[draw, diamond, fill=yellow!20, minimum width=4cm, aspect=2]
```

Finally well define a style for the arrows. For this we set the line thickness to thick, add an arrow head and specify the stealth arrow head.

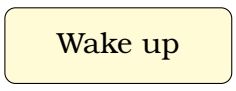
```
\tikzstyle{arrow} = [thick,->,>=stealth]
```

6.2.2 Nodes

To add a node we use the \node command. We then add a label for the node in parenthesis. This label is how we refer to the node in the rest of the code. Then in square brackets we add the name of the tikzstyle we want the node to conform to, along with any other formatting options. Then in curly brackets we add the text we want to appear in the block before closing the statement with a semicolon.

If you use following code with above defined tikz command you will see right hand side figure. Here node name is walking and its position is $(0, 0)$ point.

```
\begin{tikzpicture}
\tikzstyle{process}=[draw,rectangle, rounded corners,
fill=yellow!20, minimum width=3cm, minimum height=1cm]

\tikzstyle{decision}=[draw, diamond, fill=red!20,
minimum width=4cm, aspect=2]

\tikzstyle{arrow}=[thick,->,>=stealth]
\node[process] (waking) at (0,0)  {Wake up};
\end{tikzpicture}
```



Now we are ready to build our flow chart. At different point we need to create different node as our choice.

6.2.3 Arrow

To finish off our flowchart we need to add the arrows in. To draw an arrow we use the `\draw` command and then specify the tikzstyle we prepared for arrows using square brackets. We then enter the label of the node we want the arrow to start from, followed by two dashes and then the label corresponding to the node we want the arrow to terminate at. The labels need to be in parenthesis and we need to make sure we close the statement with a semicolon.

We use square brackets immediately after the keyword node and then enter anchor= followed by the anchor.

```
\draw[arrow] (waking) -- (day);
\draw[arrow] (day) -- node[anchor=east] {Yes} (up);
\draw[arrow] (day) --node[anchor=south] {No}  (sleep);
\draw[arrow] (sleep) |- (waking);
```

Bellow is the whole code of above flowchart defined in the beginning.

```
\begin{tikzpicture}
\tikzstyle{process}=[draw,rectangle, rounded corners, fill=yellow!20,
                        minimum width=3cm, minimum height=1cm]

\tikzstyle{decision}=[draw, diamond, fill=red!20, minimum width=4cm, aspect=2]

\tikzstyle{arrow}=[thick,->,>=stealth]

\node[process] (waking) at (0,0) {Wake up};

\node[decision] (day) at (0,-3) {Is it a weekday?};

\node[process] (up) at (0,-6) {Get out of bed};
\node[process] (sleep) at (5,-3) {Go back to sleep};

\draw[arrow] (waking) -- (day);
\draw[arrow] (day) -- node[anchor=east] {Yes} (up);
\draw[arrow] (day) --node[anchor=south] {No}  (sleep);
\draw[arrow] (sleep) |- (waking);

\end{tikzpicture}
```

6.2.4 Example I

Before that let us define few more tikzstyle. First lets define the block were going to use for start and stop blocks.

```
\tikzstyle{startstop} = [rectangle, rounded corners, minimum width=3cm,
        minimum height=1cm, text centered, draw=black, fill=red!30]
```

Next well specify an input or output box. This time we want the block to be a parallelogram. To achieve this we ask for a trapezium and then alter the angles. The rest is very similar.

```
\tikzstyle{io} = [trapezium, trapezium left angle=70, trapezium right angle=110,
minimum width=3cm, minimum height=1cm, text centered, draw=black, fill=blue!30]
```

We can define node in a very compact way. First define a node without any position vector. For example

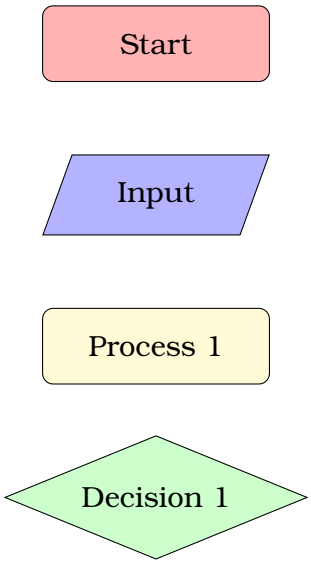
```
\node (start) [startstop] {Start};
```

We will define distance between nodes using `node distance=2cm` which will build the blocks are automatically spaced 2cm apart from their centres.

```
\begin{tikzpicture}[node distance=2cm]

<TikZ code>

\end{tikzpicture}
```

Now onwards to define node we need to tell the node where to position itself. To do this we enter below of followed by an equals sign and a node label into the square brackets. We could also use above of, right of or left of if we wanted the block to appear somewhere else. Well tell it to position itself below the start block.

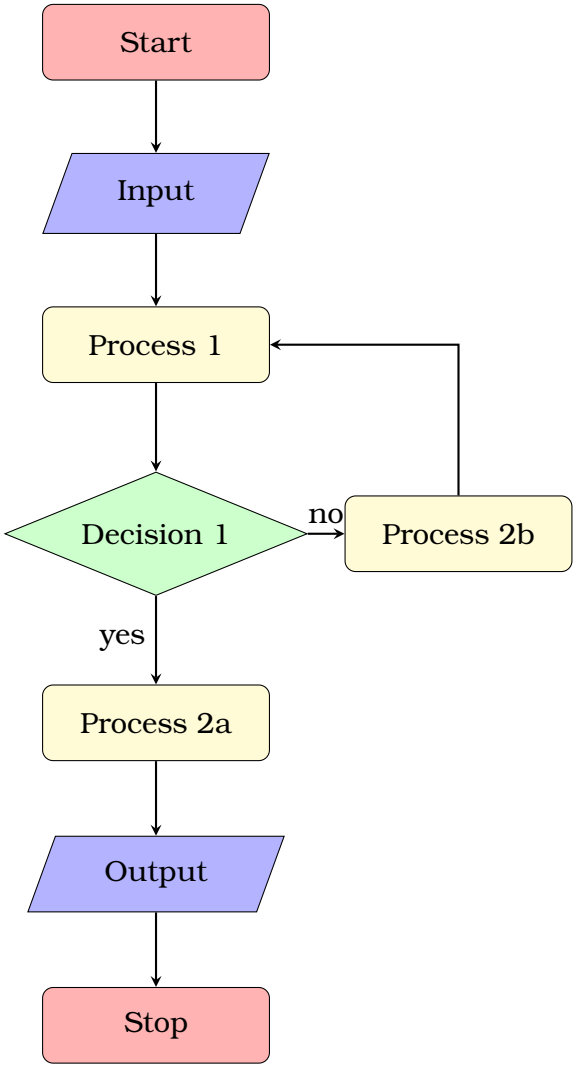
```
\begin{tikzpicture}[node distance=2cm]
\tikzstyle{process}=[draw,rectangle, rounded corners,
fill=yellow!20, minimum width=3cm,
minimum height=1cm]

\tikzstyle{decision}=[draw, diamond, fill=green!20,
minimum width=4cm, aspect=2]

\tikzstyle{startstop} = [rectangle, rounded corners,
minimum width=3cm, minimum height=1cm,
text centered, draw=black, fill=red!30]

\tikzstyle{io} = [trapezium, trapezium left angle=70,
trapezium right angle=110, minimum width=3cm,
minimum height=1cm, text centered, draw=black,
fill=blue!30]

\tikzstyle{arrow}=[thick,->,>=stealth]

\node (start) [startstop] {Start};
\node (in1) [io, below of=start] {Input};
\node (pro1) [process, below of=in1] {Process 1};
\node (dec1) [decision, below of=pro1] {Decision 1};
\end{tikzpicture}
```



If we compile the code youll notice that the gap between the green decision block and the orange process block isnt as big as the other gaps. This is because the decision block, being a diamond, is taller than the other blocks. Therefore we can manually

adjust its position using the `yshift` variable. (We can also use `xshift` variable) If we enter `yshift=-0.5cm` it will move the decision block vertically down by $0.5cm$ which should make the gap more regular. Using this fact we can make our flowchart more beautiful. Add the following code with above code.

```
\node (dec1) [decision, below of=pro1,
yshift=-0.5cm] {Decision 1};

\node (pro2a) [process, below of=dec1,
yshift=-0.5cm] {Process 2a};

\node (pro2b) [process, right of=dec1,
xshift=2cm] {Process 2b};

\node (out1) [io, below of=pro2a] {Output};

\node (stop) [startstop, below of=out1] {Stop};

\draw [arrow] (start) -- (in1);
\draw [arrow] (in1) -- (pro1);
\draw [arrow] (pro1) -- (dec1);
\draw [arrow] (pro2b) |- (pro1);

\draw [arrow] (dec1) -- node[anchor=east]
                        {yes} (pro2a);
\draw [arrow] (dec1) -- node[anchor=south]
                        {no} (pro2b);
\draw [arrow] (pro2a) -- (out1);
\draw [arrow] (out1) -- (stop);
```



6.2.5 Text width

The final thing we should discuss is the text width. At the moment all our text fits nicely inside our shapes. However, if for example, we add some more text to process 2a, youll see the shape just extends horizontally until the text fits.

```
\node (pro2a) [process, below of=dec1, yshift=-0.5cm] {Process 2a text text text text text
        text text text text};
```

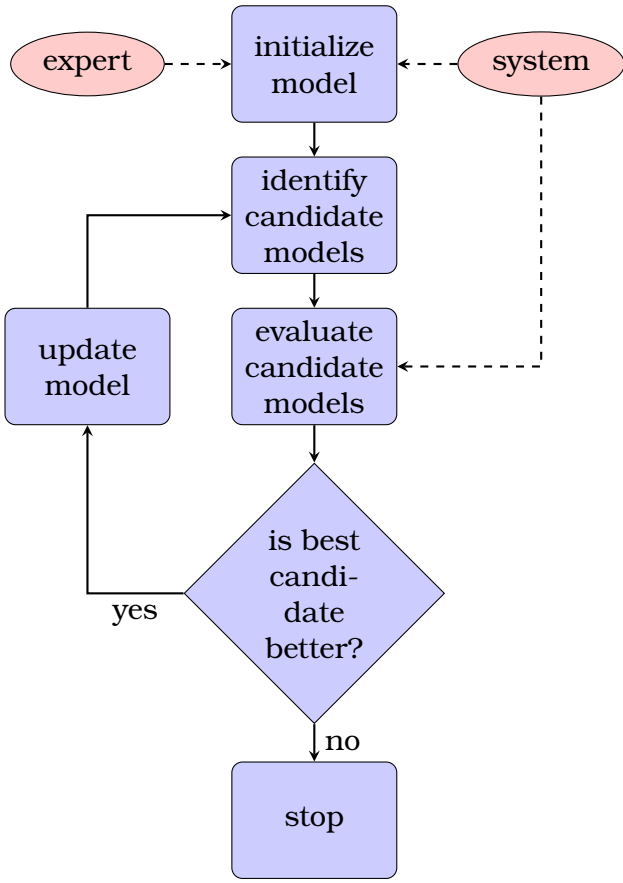
To improve it we can specify the text width for these nodes by entering `text width=` followed by a length into our `tikzstyles`.

```
\tikzstyle{process} = [rectangle, minimum width=3cm, minimum height=1cm, text centered,
text width=3cm, draw=black, fill=orange!30]
```

6.2.6 Example II

Here is another complete example.

```
\begin{tikzpicture}[node distance = 2cm, auto]

% Define block styles
\tikzstyle{decision}=[diamond,draw,fill=blue!20,
 text width=4.5em, text badly centered,
 node distance=3cm, inner sep=0pt]
\tikzstyle{block} = [rectangle, draw,
fill=blue!20,  text width=5em, text centered,
rounded corners, minimum height=4em]
\tikzstyle{line} = [thick,->,>=stealth]
\tikzstyle{cloud} = [draw, ellipse,fill=red!20,
node distance=3cm,        minimum height=2em]

% Place nodes
\node [block] (init) {initialize model};
\node [cloud, left of=init] (expert) {expert};
\node [cloud, right of=init] (system) {system};
\node [block, below of=init] (identify)
{identify candidate models};
\node [block, below of=identify] (evaluate)
{evaluate candidate models};
\node [block, left of=evaluate,
node distance=3cm] (update) {update model};
\node [decision, below of=evaluate] (decide)
{is best candidate better?};
\node [block, below of=decide,
node distance=3cm] (stop) {stop};
% Draw edges
\draw [line] (init) -- (identify);
\draw [line] (identify) -- (evaluate);
\draw [line] (evaluate) -- (decide);
\draw [line] (decide) -| node [near start]
                                {yes} (update);
\draw [line] (update) |- (identify);
\draw [line] (decide) -- node {no}(stop);
\draw [line,dashed] (expert) -- (init);
\draw [line,dashed] (system) -- (init);
\draw [line,dashed] (system) |- (evaluate);
\end{tikzpicture}
```

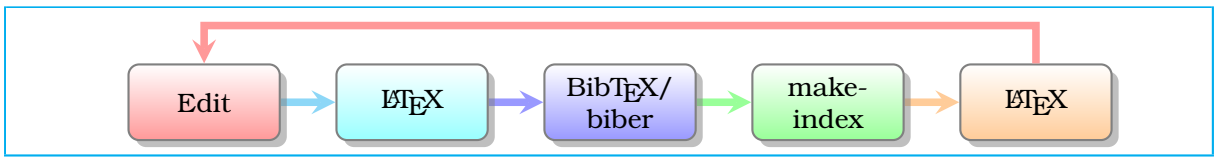


6.3 Smartdiagram Package

The `smartdiagram` package makes building diagrams of various types very easy. You need to load the `smartdiagram` package: `\usepackage{smartdiagram}` before defining `\begin{document}`. Define the diagram. An option in square brackets defines the type, and an argument in curly braces contains a comma-separated list of items: The following simple code gives us nice look.

```
\smartdiagram[flow diagram:horizontal]{Edit,\LaTeX, Bib\TeX/ biber, make\-index,
                                                                    \LaTeX}
```

The following figure shows the horizontal diagram:



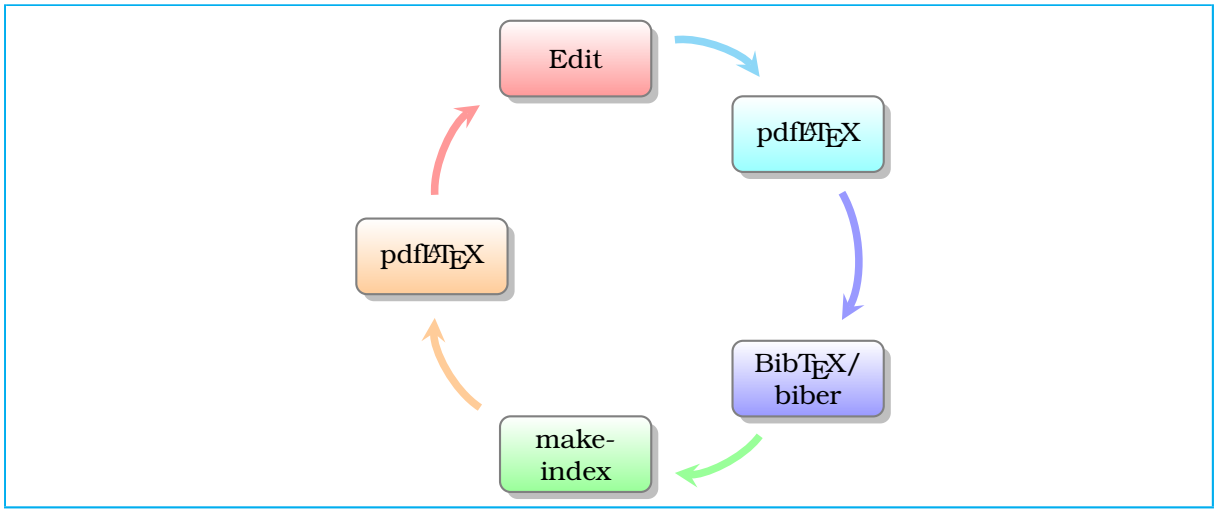
A simple call of `\smartdiagram` plus arguments produced the image. The syntax of the command is as follows:

```
\smartdiagram[type of diagram]{list of items}
```

Lets try a circular diagram. It is counterclockwise by default. Add :clockwise to the option to get a clockwise order:

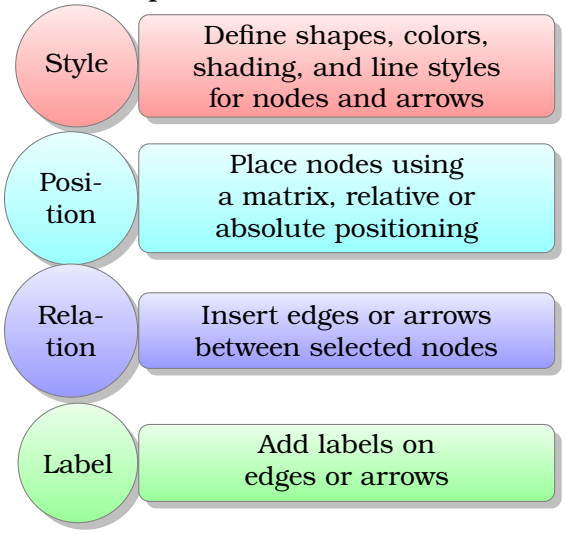
```
\smartdiagram[circular diagram:clockwise]{Edit, pdf\LaTeX, Bib\TeX/ biber, make\-i
```

The following figure shows the circular diagram:



For nicely arranging items with a description, theres the descriptive diagram. The item is a list of small lists, as a consequence. We use additional curly braces to hide the comma within an item so that it wont be taken as an item separator:

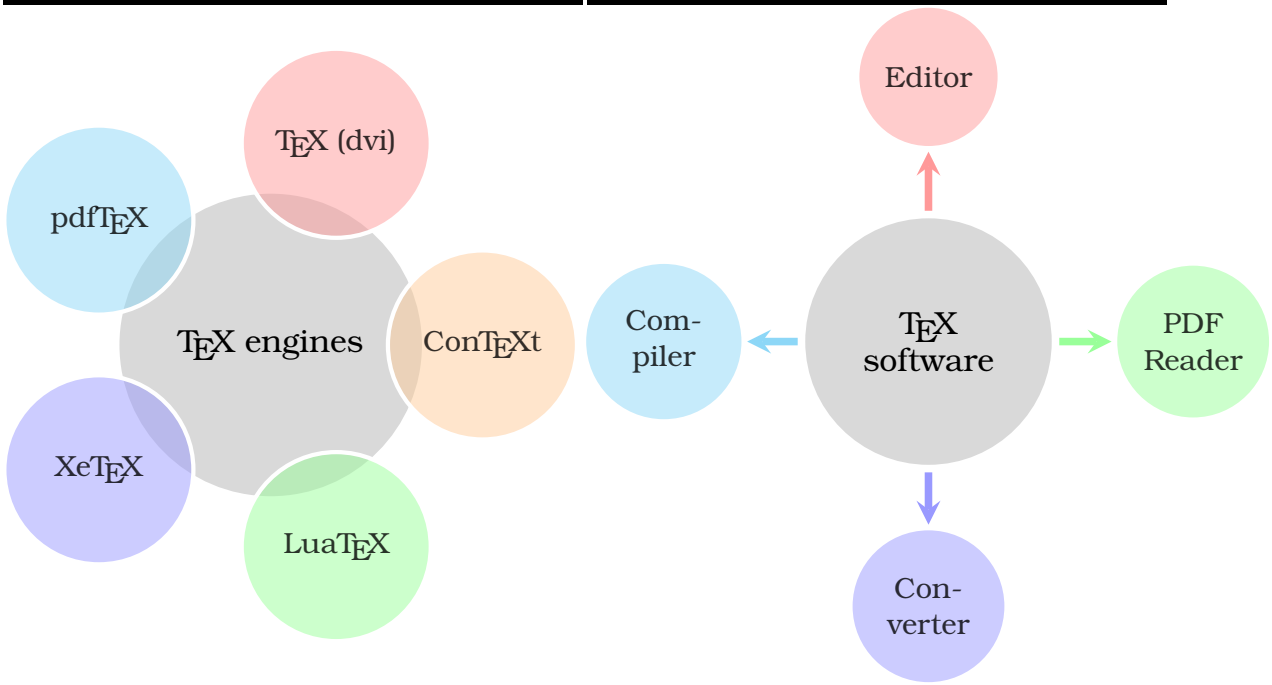
```
\smartdiagram[descriptive diagram]{
{Style, Define shapes, colors, shading,
and line styles for nodes and arrows},

{Position, Place nodes using a matrix,
        relative or absolute positioning},

{Relation, Insert edges or arrows
                between selected nodes},

{Label, Add labels on edges or arrows}

}
```



Following are the example of `bubble diagram` and `constellation diagram`

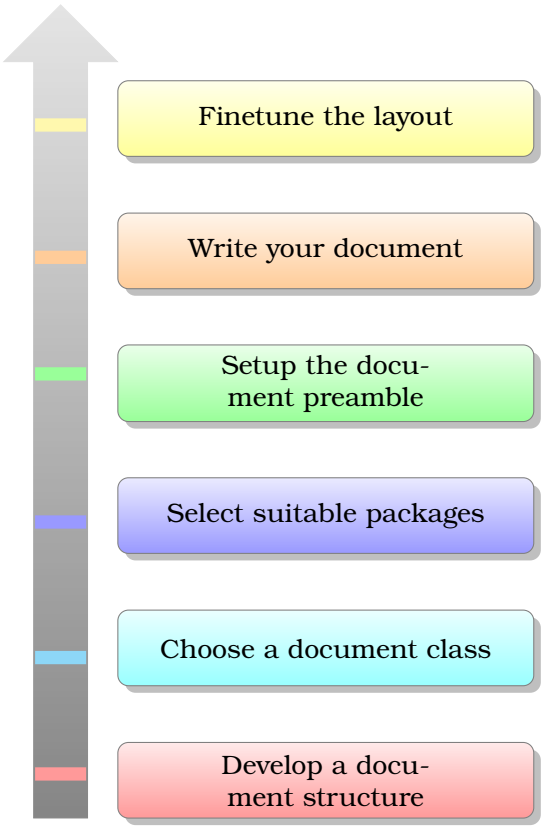
```
\smartdiagram[bubble diagram]{
\TeX\ engines, \TeX\ (dvi), pdf\TeX,
 Xe\TeX,  Lua\TeX, Con\TeX t}
```

```
\smartdiagram[constellation diagram]{
\TeX\ software, Editor, Compiler,
 Converter, PDF Reader}
```



If your descriptive diagram has a certain order and you would like to emphasize that order, use a priority descriptive diagram, like this:

```
\smartdiagram[priority descriptive diagram]{
        Develop a document structure,
        Choose a document class,
        Select suitable packages,
        Setup the document preamble,
        Write your document,
        Finetune the layout}
```



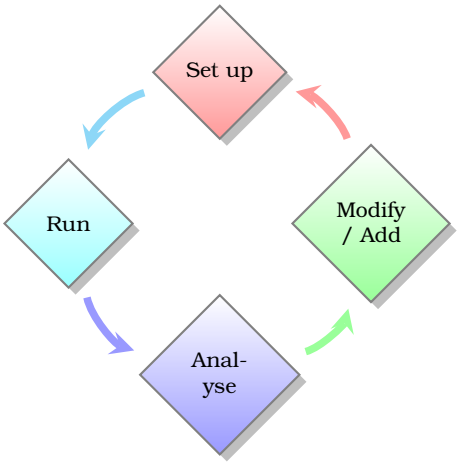
The possible diagrams that could be created are:

- `circular diagram:` the items in the list are diplayed around a circle;
- `flow diagram:` the items in the list are diplayed as a flow chart;
- `descriptive diagram:` a diagram in which are displayed concepts and their description;
- `priority descriptive diagram:` a diagram in which the items are deployted based on their relevance;
- `bubble diagram:` each item is a bubble deployted around a bubble center, which is the first element in the list;
- `constellation diagram:` each item is a circle connected to the center, the first element in the list again;
- `connected constellation diagram:` each item is a circle and, a part from the first element in the list, the other ones are connected toghether.

6.3.1 Customize smart diagram

You can customized any one of the smart diagram using `\smartdiagramset`. Here we redefine circular diagram set, circular to diamond by `module shape=` command. Also we define its minimum width, height by command `module minimum width=`, `module minimum height` respectively. By `circular distance=` we can fix distance between two module node.

```
\smartdiagramset{module shape=diamond,font=\scriptsize,
        module minimum width=1cm, module minimum height=1cm,text width=1cm,
        circular distance=2cm}

\smartdiagram[circular diagram]{Set up,Run,Analyse,Modify~/ Add}
```



One can also use `tikz` to define your own module style.

```
        \tikzset{my decoration/.style={decorate,decoration=zigzag}}

\smartdiagramset{module shape=rectangle,insert decoration={my decoration}}

        \smartdiagram[flow diagram]{Set up,Run,Analyse,Modify~/ Add}
```