My LaTeX Demos

Mustafa Can Yucel

April 29, 2016

Abstract

This is a sample document for demonstrating what I have learned during LaTeX study. There are examples for styles, floating objects, commands and almost all other commonly used LaTeX properties and settings.

The environment used is MikTeX 2.9 and TexMaker and lately TexStudio. The current incarnation LaTeX 2_ε is used as of April 29, 2016.

Contents

1 Things Needed to Know

1.1 Spaces

White-space characters, whatever their numbers are, are treated as a single space. In this sentence, there are actually six spaces between two brackets:( ). You see only one, right? Funny stuff.

An empty line in the code starts a new paragraph, such as this one.

1.2 Special Characters

The following symbols are reserved as they either have a special meaning in LaTeX, or they are not available in all the fonts: # $ % & _ { } \

These can be used in text body by the escape character backslash. Backslash cannot be escaped by itself, so the command \textbackslash is used:

```
\# \$ \% \^ \& \_ \{ \} \textbackslash
```

1.3 Comments

% is used for simple commenting:

This sentence has a comment in itself.

For longer comments, *comment* environment of verbatim package may be used.

This sentence also has a comment in it.

```
This sentence %though stupid
%but still
has a comment in itself.

For longer comments, \textsl{comment} environment of verbatim package may be used.

This sentence also has a \begin{comment} still stupid,
but longer and more
structured \end{comment}
comment in it.
```

1.4 Starting a New Page

When we want to start a new page, there are several alternatives to be used.

newpage This command immediately ends the page, but does not process float queues. Therefore, if you have any floats in these lists, they will be put to the new page.

clearpage It ends the current page and causes all figures and tables appeared so far in the input to be printed.

cleardoublepage This also ends the current page and causes all figures and tables appeared so far in the input to be printed. Moreover, in a two-sided printing style, it also makes the new page a right-hand (an odd-numbered) page, producing a blank page if necessary.

2 Typesetting Text

2.1 Quotation Marks

This is an example sentence to demonstrate 'single' and "double" quotation marks. Opening mark is done by grave accent and closing mark is done by vertical quote:

```
'single' vs ''double''
```

2.2 Degree Symbol

Creating °C by pure LaTeX is complex. Creating ℃ by `textcomp` package is easy, though:

```
$\,^{\circ}\mathrm{C}$ versus \textcelsius
```

2.3 Ellipsis

On a typewriter, period and comma have the same width with other characters. On a print, however, they are smaller and closer to the preceding character. Therefore, using three dots for 'ellipsis' is not correct. The command *\ldots* should be used:
This is ... not true.
This is … true.

```
This is ... not true.\\
This is \ldots true.
```

2.4 Cross-References

label, ref, pageref commands are used for cross referencing with a user defined marker string. LaTeX replaces *ref* by the number of the section, subsection, figure, table or theorem after which the corresponding *label* command occurred. Another good point of LaTeX is that you can easily reference almost anything that is numbered, and LaTeX will take care of the numbering, updating it when necessary. The commands do not depend on environment to be referenced, it is just:

```
\label{marker}
...
\ref{marker}
```

The label `marker` will not be visible in the compiled product. LaTeX will calculate the correct number for the objects in the document, and then it will replace the string `\ref{marker}` with the right number that was assigned to the object. If a reference to a marker that does not exist is used, the document may or may not be compiled (TexStudio gives an error, but other programs may compile just giving a warning.). If compiled, unknown references will be shown as "??" in the PDF. As you may have guessed, compiling references is a two step process; first the compiler has to store the labels with the right numbering to be used for referencing. And then it has to replace all `\ref` strings with the

correct numbers. That's why compiling a document twice is required to update all the references when large changes are done in reference list. Some IDEs do this automatically (such as TexStudio), as if LaTeX realizes references may have changed to require a second compilation, it throws a warning:

```
LaTeX Warning: Label(s) may have changed. Rerun to get
cross-references right.
```

Using the command `\pageref` returns the page of the referenced marker. A reference to this sub-section looks like: "see section 2.4 on page 5".

```
see section \ref{this} on page \pageref{this}
```

Since the same command is used for every environment, it is customary to use conventional abbreviations for different types of labels:

fig:	Figure
sec:	Section
subsec:	Subsection
chap:	Chapter
tab:	Table
eq:	Equation
lst:	Code Listing
itm:	Enumerated List Item
app:	Appendix Subsection

Some packages such as *fancyref* also rely on these meta data information. Another suggestion is to avoid numbering in label identifiers, as re-numbering will be required on most of the changes.

Floats such as `table` or `figure` are NOT label-able. Therefore, they need a *caption* element as an identifier. While referencing these environments, `\label` command should be put after the `\caption` element. To be completely safe, it could be put inside the `\caption` element as well. If not put inside or after captions, reference commands will pick up the current section or list number instead of what you intended.

In case you use the package *hyperref* to create a PDF, the links to tables or figures will point to the caption of the table or figure, which is always below/above the table or figure itself. Therefore the table or figure will not be visible, if it is above the pointer and one has to scroll up in order to see it. If you want the link point to the top of the image you can give the option `hypcap` to the *caption* package:

```
\usepackage[hypcap]{caption}
```

Referencing equations is the same:

```
\begin{equation} \label{eq:solve}
x^2 - 5 x + 6 = 0
\end{equation}
\begin{equation}
x_1 = \frac{5 + \sqrt{25 - 4 \times 6}}{2} = 3
\end{equation}
```

```
\begin{equation}
x_2 = \frac{5 - \sqrt{25 - 4 \times 6}}{2} = 2
\end{equation}

and so we have solved equation ~\ref{eq:solve}
```

The *amsmath* dds a new command for referencing formulae; it is `\eqref`. 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 formulae and other things, without the need to repeat the word "formula" before any reference. Its output can be changed as desired; for more information see the *amsmath* documentation.

The *amsmath* package also adds the `\numberwithin{countera}{counterb}` command which 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 numbers.

The *cases* package adds the `\numcases` and the `\subnumcases` commands, which produce multi-case equations with a separate equation number and a separate equation number plus a letter, respectively, for each case.

The *hyperref* package introduces another useful command; `\autoref{}`. This command creates a reference with additional text corresponding to the target's type, all of which will be a hyperlink. For example, the command `\autoref{sec:intro}` would create a hyperlink to the `\label{sec:intro}` command, wherever it is. Assuming that this label is pointing to a section, the hyperlink would contain the text "section 3.4", or similar.

When you define a `\label` outside a figure, a table, or other floating objects, the label points to the current section. In some cases, this behavior is not what you'd like and you'd prefer the generated link to point to the line where the `\label` is defined. This can be achieved with the command `\phantomsection` as in this example:

```
%The link location will be placed on the line below.
\phantomsection
\label{the_label}
```

2.5 Footnotes

The command *footnote* is used. As an example this sentence[1] has a footnote.

2.6 Emphasizing Words

For the aim of emphasize, words can be underlined or *emphasized*[2]:

```
\underline{underlined}
\emph{emphasized}
```

[1] Yes, this one

[2] Does not make sense, does it?

2.7 Environments

Environments are defined between *begin* and *end* commands.

2.7.1 Itemize, Enumerate, Description

itemize is used for simple lists, *enumerate* for enumerated lists and *description* for definitions:

1. List environments can be nested in any fashion:
   - Still, after some level it looks silly
   - with some special characters
2. Therefore remember:

   Stupid things will not be smart because they are in a list.

   Smart things can be presented beautifully in lists.

```
\begin{enumerate}
\item List environments can be nested in any fashion:
\begin{itemize}
\item Still, after some level it looks silly
\item[-] with some special characters
\end{itemize}
\item Therefore remember:
\begin{description}
\item[Stupid] things will not be smart because they are in a list.
\item[Smart] things can be presented beautifully in lists.
\end{description}
\end{enumerate}
```

2.7.2 Flushleft, Flushright and Center

They generate paragraphs of left, right, or center aligned.

This text
is left-aligned. Each line length is not equal.

This text is
right-aligned. Each line is also not justified.

This is
the center.

2.7.3 Quote, Quotation and Verse

These can be useful in many ways.

> Quote is for short sentences.

Quotation supports paragraph first line indentation, therefore better suited for longer ones.

verse is for
poems, for line
breaks are
important

3 Floating Bodies

As our publications include *a lot of* figures and tables, floating bodies are one of the most important points of LaTeX. These elements need special attention, since they cannot be broken across pages. As you cannot start a new page every time there is a figure or table (this will result in blank spaces everywhere), you should 'float' these items whenever they do not fit on the current page, while filling the current page with body text. There are two different environments for floating bodies; one for *figures* and one for *tables*. It is very important to understand how LaTeXhandles floats; otherwise they will be a major pain in the back.

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes `table` and `igure` floats, but you can define new ones of your own. Floats are there to deal with the problem of the object that won't fit on the present page, and to help when you really don't want the object here just now.

Floats are not part of the normal stream of text, but separate entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the designer specifies). They always have a caption describing them and they are always numbered so they can be referred to from elsewhere in the text. LaTeX automatically floats Tables and Figures, depending on how much space is left on the page at the point that they are processed. If there is not enough room on the current page, the float is moved to the top of the next page. This can be changed by moving the Table or Figure definition to an earlier or later point in the text, or by adjusting some of the parameters which control automatic floating.

Authors sometimes have many floats occurring in rapid succession, which raises the problem of how they are supposed to fit on the page and still leave room for text. In this case, LaTeX stacks them all up and prints them together if possible, or leaves them to the end of the chapter in protest. The skill is to space them out within your text so that they intrude neither on the thread of your argument or discussion, nor on the visual balance of the typeset pages.

3.1 Placement

LaTeX will place every float in encounters according to the placement specifier supplied by the author. If a float cannot be placed on the current page, it is deferred either to *figures* queue or *tables* queue[3]. When a new page is started, LaTeX first checks if it is possible to fill a special 'float' page with floats from the queues. If this is not possible, the first float on each queue is treated as if it has just occurred in the text: LaTeX again tries to put it according to its respective placement modifiers. Any new floats occurring in the text get placed into the

[3]These are FIFO queues

appropriate queues. LaTeX strictly maintains the original order of the appearance of each type of float. Therefore, a figure that cannot be placed pushes all further figures to the end of the document. Therefore:

> If LaTeX is not placing the floats as expected, it is often only one float jamming one of the two float queues.

While it is possible to give single placement specifiers, this causes potential problems as if the float cannot be placed on the location specified, it becomes stuck, blocking the subsequent floats. It is advised to never use the [h] option; in new versions it is automatically replaced by [ht].

Any material enclosed within a `table` or `figure` environment will be treated as a floating matter. Both support an optional parameter, *placement specifier*. This is used to tell the locations which the float is allowed to be moved. A *placement specifier* is constructed by building a string of *float-placing permissions*:

Spec	Permission
h	here at the very place in the text where it occured. For small floats.
t	at the *top* of a page
b	at the *bottom* of a page
p	on a special *page* containing only floats
!	without considering parameters, could stop float from being placed!

3.2 Table

Tables are important, and sadly a little more complex than they should be in LaTeX.
The arguments are as follows:

\begin{tabular}[pos]{table spec}

table spec Defines the format of the table:

- l left aligned
- r right aligned
- c centered text
- p{width} justified text
- — vertical line

pos Vertical position of the table relative to baseline of surrounding text:

- t top
- b bottom
- c center

7C0	hexadecimal
3700	octal
1110000	binary
1984	decimal

Within the *tabular* environment, & jumps to the next column, \\starts a new line, and \hline inserts a horizontal line. Partial lines are extended using command \cline{i-j} where i and j are column numbers the line should extend over:

The code-behind of this table is:

```
\begin{table}[htbp]
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimal \\
3700 & octal \\ \cline{2-2}
1110000 & binary \\
\hline \hline
1984 & decimal \\
\hline
\end{tabular}
\end{table}
```

It is usually an easier approach to create a "sketch" table using the wizards of GUI programs (such as *Quick Tabular* in TexMaker, and then fine tune the computer-generated code then code everything from scratch.

It should be noted that the default tables in LaTeXare not stretched for line width, and look primitive. There are many solid packages for table beautification, but they require extensive coding (though it may be possible to find pre-generated formats in Internet). The two important packages that should be implemented in every table for ease of view and basic aesthetics are `tabularx` and `booktabs`. The first one enables stretching tables and the second one improves view with their new commands.

3.2.1 tabularx and booktabs Packages

`tabularx` package has a new column type X which enables stretching, whereas `booktabs` gives new horizontal line options.

The common consensus on Internet regarding table aesthetics is that vertical rules (lines, in British typesetting terminology) should be avoided at all costs, as well as double rules. Also, horizontal rules should not be repeated at every row; a table basically needs three rules: two thick ones at top and bottom, and a thin one after the heading row. These are supplied by the `booktabs` package with the commands \toprule, \midrule, and \bottomrule. Their widths (or thicknesses) can be adjusted manually if required.

The following code creates the table given in

```
\begin{table}
\centering
\caption{Parameter Sets and Their Values}
```

Table 1: Parameter Sets and Their Values

	SET 1	SET 2	SET 3	SET 4
n1	4.0	4.0	7.0	7.0
n2	5.5	5.5	8.5	8.5
n3	8.5	8.5	11.5	11.5
alpha	0.1	0.2	0.1	0.2
K	4000	4000	4000	4000
dy	0.015	0.015	0.015	0.015

```
\label{tab:fdParameterSets}
\begin{tabularx}{\textwidth}{XXXXX}
\toprule & SET 1 & SET 2 & SET 3 & SET 4 \\
\midrule
n1 & 4.0 & 4.0 & 7.0 & 7.0 \\
n2 & 5.5 & 5.5 & 8.5 & 8.5 \\
n3 & 8.5 & 8.5 & 11.5 & 11.5 \\
alpha & 0.1 & 0.2 & 0.1 & 0.2 \\
K & 4000 & 4000 & 4000 & 4000 \\
dy & 0.015 & 0.015 & 0.015 & 0.015 \\
\bottomrule
\end{tabularx}
\end{table}
```

More finer tweaks can also be made using different commands:

```
\begin{table}
\centering
\caption{Member End Displacement Equation Signs}
\label{tab:columnEndDisplacementSigns}
\begin{tabularx}{\textwidth}{XXXX}
\toprule \centering \rule[-2ex]{0pt}{5.5ex} Point & \centering  $\overrightarrow{r}$
    & \centering $\overrightarrow{\theta} \times \overrightarrow{r}$ & Displacement Equati
\midrule \centering \rule[-2ex]{0pt}{5.5ex} A & \centering  $\{-1, 1, 0\}$
    & \centering $\{-1, -1, 0\}$ & $\Delta_{xi} = \delta_{xc} - \theta \cdot e_{yi}$ \newl
\centering \rule[-2ex]{0pt}{5.5ex} B & \centering  $\{1, 1, 0\}$  & \centering $\{-1, 1, 0
    &  $\Delta_{xi} = \delta_{xc} - \theta \cdot e_{yi}$ \newline  $\Delta_{yi} = \delta_{
\centering \rule[-2ex]{0pt}{5.5ex} C & \centering  $\{1, -1, 0\}$ & \centering $\{1, 1, 0\
\centering \rule[-2ex]{0pt}{5.5ex} D & \centering  $\{-1, -1, 0\}$  & \centering $\{1, -1,
\bottomrule
\end{tabularx}
\end{table}
```

3.3 Figures

To create a figure that floats, use the `Figure` environment. `\caption{caption text}` is used to define a caption for the float. A running number and the string "Figure" or "Table" will be added by LaTeX.

Table 2: Member End Displacement Equation Signs

Point	$\overrightarrow{r}$	$\overrightarrow{\theta} \times \overrightarrow{r}$	Displacement Equation
A	$\{-1, 1, 0\}$	$\{-1, -1, 0\}$	$\Delta_{xi} = \delta_{xc} - \theta \cdot e_{yi}$ $\Delta_{yi} = \delta_{yc} - \theta \cdot e_{xi}$
B	$\{1, 1, 0\}$	$\{-1, 1, 0\}$	$\Delta_{xi} = \delta_{xc} - \theta \cdot e_{yi}$ $\Delta_{yi} = \delta_{yc} + \theta \cdot e_{xi}$
C	$\{1, -1, 0\}$	$\{1, 1, 0\}$	$\Delta_{xi} = \delta_{xc} + \theta \cdot e_{yi}$ $\Delta_{yi} = \delta_{yc} + \theta \cdot e_{xi}$
D	$\{-1, -1, 0\}$	$\{1, -1, 0\}$	$\Delta_{xi} = \delta_{xc} + \theta \cdot e_{yi}$ $\Delta_{yi} = \delta_{yc} - \theta \cdot e_{xi}$

The two commands `\listoffigures` and `\listoftables` operate similar to table of contents; printing a list of figures or tables. These lists will display the whole caption text, therefore the short versions could be used by a optional parameter as follows:
`\caption[short]{Looooooooooooooooong}`

Figure 3 is an example for both how to use image place holders and cross-references.

It's possible to get a thin border around all figures. You have to write the following once at the beginning of the document:

```
\usepackage{float}
\floatstyle{boxed}
\restylefloat{figure}
```

The border will not include the caption.

The following is a combination of `Figure` and `Table` environments:

Figure 1: A picture of a gull

```
\begin{figure}[htbp]
```

Figure 2: Same gull looking other way

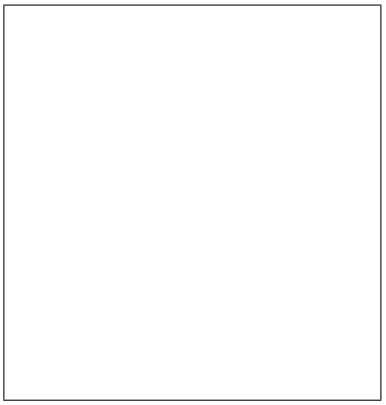


Figure 3: Five by five in centimeters.

1	2	3
4	5	6
7	8	9

Table 3: A simple table

```
\caption{A picture of a gull}
\centering
\includegraphics[width=0.5\textwidth]{images/gull}
\end{figure}
\begin{figure}[htbp]
\centering
\reflectbox{
\includegraphics[width=0.5\textwidth]{images/gull}}
\caption{Same gull looking other way}
\end{figure}
\begin{figure}[htbp]
\makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
\caption{Five by five in centimeters.\label{placeholder}}
\end{figure}
\begin{table}[h!]
\begin{center}
\begin{tabular}{| l c r |}
\hline
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\hline
\end{tabular}
\end{center}
\caption{A simple table}
\end{table}
```

3.4 Subfloats

A useful extension is *subcaption* package which enables sub-floats inside a single float. The *subfigure* and *subfig* packages are no longer maintained and deprecated, but they are useful alternatives when used in conjunction with LaTeX templates that are not compatible with *subcaption*. If you intend to cross-reference any of the subfloats, see where the label is inserted; `\caption` outside the subfigure-environment will provide the global caption.

subcaption will arrange the figures or tables side-by-side providing they can fit, otherwise, it will automatically shift subfloats below. This effect can be added manually, by putting the newline command (\\) before the figure you wish to move to a newline.

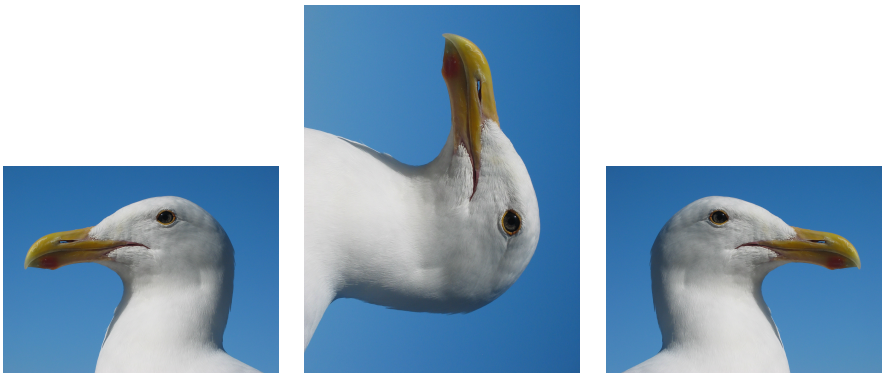
Horizontal spaces between figures are controlled by one of several commands, which are placed in between `\begin{subfigure}` and `\end{subfigure}`:

- A non-breaking space (specified by tilde) can be used to insert a space in

between the subfigs.

- Math spaces: \qquad, \quad, \;, and \,
- Generic space: \hspace{''length''}
- Automatically expanding/contracting space: \hfill

An example:



(a) Left? (b) Top? (c) Right?

Figure 4: Same Gull Different Directions (Same Image)

```
\begin{figure}[htbp]
\centering
\begin{subfigure}[b]{0.3\textwidth}
\includegraphics[width=\textwidth]{images/gull}
\caption{Left?}
\label{fig:gullLeft}
\end{subfigure}
%add desired spacing between images, e. g. ~, \quad, \qquad, \hfill etc.
%(or a blank line to force the subfigure onto a new line)
~
\begin{subfigure}[b]{0.3\textwidth}
\includegraphics[height=\textwidth, angle=-90]{images/gull}
\caption{Top?}
\label{fig:gullTop}
\end{subfigure}
%add desired spacing between images, e. g. ~, \quad, \qquad, \hfill etc.
%(or a blank line to force the subfigure onto a new line)
~
\begin{subfigure}[b]{0.3\textwidth}
\reflectbox{
\includegraphics[width=\linewidth]{images/gull}}
\caption{Right?}
\label{fig:gullRight}
\end{subfigure}
\caption{Same Gull Different Directions (Same Image)}
```

```
\label{fig:gullAndDirections}
\end{figure}
```

If you are writing a document using two columns (i.e. you started your document with something like `\documentclass[twocolumn]{article}`), you might have noticed that you can't use floating elements that are wider than the width of a column (using a LaTeX notation, wider than `0.5\textwidth`), otherwise you will see the image overlapping with text. If you really have to use such wide elements, the only solution is to use the "starred" variants of the floating environments, that are figure* and table*. Those "starred" versions work like the standard ones, but they will be as wide as the page, so you will get no overlapping.

A bad point of those environments is that they can be placed only at the top of the page or on their own page. If you try to specify their position using modifiers like b or h they will be ignored. Add `\usepackage{dblfloatfix}` to the preamble in order to alleviate this problem with regard to placing these floats at the bottom of a page, using the optional specifier [b]. Default is [tbp]. However, h still does not work.

To prevent the figures from being placed out-of-order with respect to their "non-starred" counterparts, the package *fixltx2e* should be used.

4 Typesetting Mathematical Formulae

Mathematical formulae typesetting is one of the most powerful areas of LaTeX. The main bundle used is $\mathcal{A}_{\mathcal{M}}\mathcal{S}$LaTeX. It is a collection of packages and classes for mathematical typesetting. It is produced by *American Mathematical Society* and the most commonly used package is *amsmath*.

4.1 Single Equations

A mathematical formula can either be put in-line with a paragraph *(text style)*, or the paragraph can be broken and the formula is typeset separately *(display style)*. Mathematical equations *within* a paragraph are entered between $ and $:

Add a squared to b squared to get c squared. Or using fancy symbols: $a^2 + b^2 = c^2$

For displaying equations, the amsmath environment *equation* is used. Then \label can be used for equation number and be referred somewhere else in the text using \eqref command. We can also name the equations using \tag.

Add a squared to b squared to get c squared. Or in a fancy way;

$$a^2 + b^2 = c^2 \tag{1}$$

Einstein said

$$E = mc^2 \tag{2}$$

He did not say

$$1 + 1 = 3 \tag{dumb}$$

this is a reference to first equation, which is (2)

If the equation is not to be numbered, starred version of *equation* environment is used:

$$a^2 + b^2 = c^2$$

In text style, the height of mathematical expressions may make line spacing irregular. This can be prevented by using \smash command: A $d_{e_{e_p}}$ mathematical expression followed by a $h^{i^{g^h}}$ expression. As opposed to a smashed $d_{e_{e_p}}$ expression followed by a $h^{i^{g^h}}$ expression.

4.2 Math Mode

There are several important differences between *math mode* and *text mode* texts:

1. Most spaces and line breaks do not have any significance. Space should be specified with \, , \quad, \qquad.
2. Empty lines are not allowed.
3. Each letter is considered to be the name of a variable and will be typeset as such. If we want to put normal text within a formula, we should use \text{...} command.

4.3 Blocks of Formulae

Greek Letters are defined by their name, as case sensitive: \gamma γ vs \Gamma Γ.

Exponents, Superscripts and Subscrips are defined by ând _ characters. They only act on the next character, so groups must be combined using curly brackets:

$$p_{ij}^{3} \qquad m_{\mathrm{Knuth}} \qquad \sum_{k=1}^{3} k$$

Square root is entered as \sqrt[root]{...}. For second root, parameter is not necessary. For just the symbol, \surd is used:

$$\sqrt{x^2+\sqrt{y}} \qquad \sqrt[8]{x+p} \qquad \surd$$

Dots can be used by \cdot, \ldot, \vdot, \ddot for centre, lower, vertical and diagonal single dot. \cdots, \ldots, \vdots, \ddots are used for centre, lower, vertical and diagonal three dots.

Horizontal braces can be put using \overbrace and \underbrace commands:

$$\underbrace{\overbrace{a+b+c}^{6}\cdot\overbrace{d+e+f}^{9}}_{\text{meaning of life}} = 42$$

Vectors could be put by \vec, \overrightarrow and \overleftarrow:

$$\vec{a} \qquad \vec{AB} \qquad \overrightarrow{AB} \qquad \overleftarrow{AB}$$

Functions have different typesetting, thus they are put by their own commands. Few examples are:

$$\lim_{x\to 0}\frac{\sin x}{x} = 1$$

Bracket size is automatically determined when using \left and \right commands:

$$1+\left(\frac{1}{1-x^2}\right)^3$$

There are almost infinite number of symbols, delimiters, operators, brackets, ...etc. in $\mathcal{AMS}$-LaTeX, therefore the rest will not be covered[4].

[4] I am bored on this topic. Really.

4.4 Multiple Equations

For this type of input, the most optimal way is to use *align* environment.

This environment can align columns in the equation array, based on operator &:

$$\begin{aligned} a &= b + c && (3)\\ &= d + e + f + g + h + i + j + k + \\ &+ \ l + m + n + o && (4)\\ &= p + q + r + s && (5) \end{aligned}$$

on this equation, \nonumber prevents equation numbering of the line.

4.5 Arrays and Matrices

Array environment can be used and its usage is similar to *tabular*:

$$\mathbf{X} = \begin{pmatrix} x_1 & x_2 & \dots \\ x_3 & x_4 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

It could also be used for piecewise functions, but *amsmath* package has a better environment named *cases*:

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ x & \text{if } x > 0 \end{cases}$$

where code-behind is:

```
\begin{equation*}
|x| = \begin{cases}
-x & \text{if } x<0, \\
0 & \text{if } x=0, \\
x & \text{if } x>0
\end{cases}
\end{equation*}
```

The *amsmath* package has a better environment called `matrix` which produces better results. There are six versions with different delimiters (borders): matrix (none), pmatrix (, bmatrix [, Bmatrix {, vmatrix —, Vmatrix ——. Numer of columns is not needed to be defined as with *array*:

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \qquad \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \dots & p_{mn} \end{bmatrix}$$

```
\begin{equation*}
\begin{matrix}
1 & 2 \\
```

```
3 & 4
\end{matrix}
\qquad
\begin{bmatrix}
p_{11} & p_{12} & \ldots & p_{1n} \\
p_{21} & p_{22} & \ldots & p_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
p_{m1} & p_{m2} & \ldots & p_{mn}
\end{bmatrix}
\end{equation*}
```

4.6 Phantoms

Especially when dealing with sub/superscripts before characters, extra spaces may arise. This is prevented using \phantom command:

$$ {}^{14}_{6}\mathrm{C} \qquad \text{vs} \qquad {}^{14}_{6}\mathrm{C} $$

If a lot of text like this will be used (such as isotopes) the *mhchem* package will be a better solution.

4.7 Bold Symbols

To prevent abuse, bold symbols are not encouraged in LaTeX. However, *amsbsy* package makes this much easier as it has a \boldsymbol command. This package is included by *amsmath*.

4.8 Closing Words on Math

There are endless possibilities on math and only a handful of them are demonstrated here. *Theorems, lemmas, axioms, proofs* are not included because of their irrelevance with me. A comprehensive list of mathematical symbols is also not placed since it is readily available online. In this list, there are:

- Math mode accents
- Greek letters
- Binary relations
- Binary operators
- Big operators
- Arrows
- Arrows as accents
- Delimiters
- Large delimiters
- Non-mathematical symbols
- Miscellaneous symbols

- $\mathcal{AMS}$ delimiters
- $\mathcal{AMS}$ Greek and Hebrew
- Math alphabets
- $\mathcal{AMS}$ binary operators
- $\mathcal{AMS}$ binary relations
- $\mathcal{AMS}$ arrows
- $\mathcal{AMS}$ negated binary relations and arrows
- $\mathcal{AMS}$ miscellaneous

5 Specialities

This section will include some special topics such as including EPS, index generation, bibliography management... etc.

5.1 Creating & Including Raster and Vector Images

Images, charts, and the graphs obtained in different Office suites in Windows OS can be included in LaTeX by different approaches. Images could be included directly. For MSOffice objects, if *pdflatex* compiler will be used, then printing those graphics as PDF files is the best approach, which will be discussed in next sections. Direct LaTeX, however, can only include EPS files. Exporting the related graphics in .eps format in Windows OS will be discussed in following sections. Once the file is obtained, however, adding the file to the document will be done by:

1. Generate file somehow.
2. Load *graphicx* package in the document with

   \usepackage[*driver*]{graphicx}

   where driver is the name of the "DVI to PostScript" converter program. The available drivers could be:

   - *dvips*: default when compiling with *latex* to get a DVI file.
   - *dvipdfm*: if compiling with *latex* to get a DVI that will be converted to PDF using *dvipdfm*, to view the final document with a pdf viewer.
   - *pdftex*: default if compiling with *pdflatex*.

   Still, in the latest version, the driver parameter is optional as the package takes care everything by itself, changing the driver according to the compiler used.
3. Use the command

   \includegraphics[*key=value*]{filename}

   to include the file in the document.

Most important keys will be defined in next sections.

5.1.1 Document Options

The *graphicx* package recognizes *draft* and *final* options given in the start of the document. Using *draft* will suppress the inclusion of images in the output file and will replace the contents with the name of the image file to be used.

5.1.2 Supported Image Formats

Supported formats depend on the driver that *graphicx* package is using. Using *pdflatex* is more convenient as it supports widespread formats such as PDF, PNG, and JPG. As an example, lets say that we inserted some pictures with JPG format, and successfully created a PDF file. Then we want to compile the document as a DVI file, bam! we get an error, because EPS versions of these images are not supplied.

The only format supported while compiling with *latex* is EPS. *pdflatex* can handle the following formats:

- JPG: Best choice for inserting photos.
- PNG: Since it is *lossless*, it is the best choice for diagrams (if you cannot generate a vector version) and screenshots.
- PDF: Could be used for documents, but also images as well. It supports both vector and bitmap images, still the latter is not recommended as a JPG or PNG will do the same work with much less disk space cost.
- EPS: It can be used with the help if *epstopdf* package. Depending on the installation of LaTeX, it may or may not be necessary to include it in the preamble. If it does not work, it may be necessary to include it just after the *graphicx* package. Additionally, it will require for writing permissions to the compiler: *-shell -escape*

5.1.3 Including Graphics

The syntax of the required command is as follows:

\includegraphics[*key1=value1, key2=value2,...*]{filename}

The file name is written *without* the extension. By this way, LaTeX compiler will look for any supported image format in that directory and will take the best one (EPS if the output is DVI; JPG, PNG or PDF if the output is PDF). Therefore images can be saved in multiple formats for different purposes. For example, a directory can have "diagram.pdf" for high-quality printing purposes while "diagram.png" can be used for previewing on the monitor. There are *a lot of* keys, but the most commonly used ones are given in Table 4. Some important notes:

- The order of the keys matter, e.g. rotations should be done before specifying height and/or width.
- Trim option does not work with XeLaTeX.
- Labels defined with *chemnum* package may behave unexpectedly under some options.
- Starred version of the command will work for EPS files only. It will take crop dimensions as an extra parameter.

Table 4: Most Common Keys and Their Values

Keyname	Property
width=xx	scale to specified width†
height=xx	scale to specified height†
angle=xx	rotate counter clockwise
scale=xx	scale
keepaspectratio	guess
trim=l b r t	crop image from (l)eft, (b)ottom, (r)ight, (t)op
clip	For trimming to work, clip=true must be set
page=x	For PDF files with multiple pages
resolution=x	guess again

†*Please note that only specifying either width or height will maintain the existing aspect ratio.*

5.1.4 All in Action

After this much introduction, let's see the commmands in action! Figure 5 is a PNG image with no options defined.

Figure 5: This is a chick!

But it is a big big chick! Let's scale it a little (by 0.2), as in Figure 6.

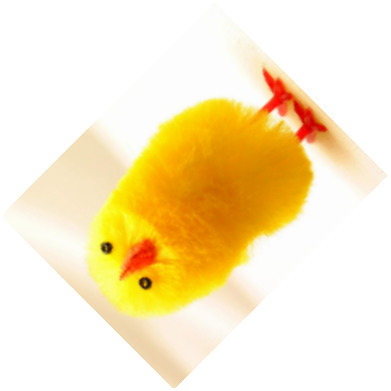
Figure 6: A smaller chick

We can also specify actual dimensions for it. Let's make its width 2.5 centimeters exactly, as in Figure 7.
We can also scale it by the width of a line in the local environment (\linewidth), width of the text on a page (\textwidth), or height of the text on a page

Figure 7: A 2.5 cm wide chick

(\textheight), yet their results are not shown here.

We can of course apply multiple operations:



Finally, we can apply cropping to focus on one particular area of interest:

Trick: Negative trim values can be used to add blank spaces to graphics, which may help in manual alignment!.

5.1.5 Creating Vector Graphics

Since it is not easy to create charts and other graphics that are not bit-map image files, some alternatives will be presented here.

Using Inkscape Direct EPS or PDF Output One of the downsides of the Microsoft Office environment is that even though a figure (an Excel chart, for example) is a vector, it cannot be exported as vector directly. For overcoming this difficulty,a multi-step approach can be used. Figure 8 is an example for a direct EPS output from Inkscape.

The procedure followed in obtaining 8 is as follows:

1. Print the graph as a PDF file.
2. Open the PDF file in Inkscape.

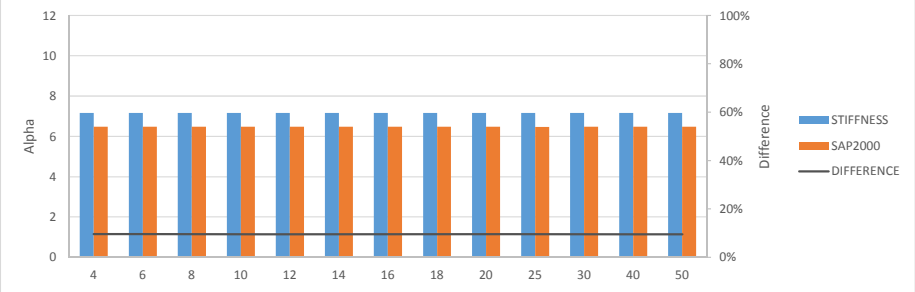


Figure 8: A "true" Vector Graph

3. Re-adjust the page size to fit to contents.
4. Save the document in Inkscape as a plain EPS or PDF file. **Do not** select any options such as *EPS+LaTeX*. Only *Export area is drawing* may be selected.
5. Include epstopdf package in preamble (`\usepackage{epstopdf}`) if an EPS file will be used.
6. Use the resultant EPS or PDF file in the document as follows:

```
\begin{figure}[htbp]
\centering
\includegraphics[width=\columnwidth]{images/inkscapeEPS}
\caption{A ``true'' Vector Graph}
\label{inkscapeEPS}
\end{figure}
```

Please note that if EPS file type will be used, *epstopdf* package will convert these to PDF files with names such as *oldFileName-eps-converted-to.pdf*. Therefore, to eliminate this one extra (though automatic) step, we may select to save PDF from Inkscape directly. In this case, the only reason to fiddle with Inkscape is to adjust the page size of the pdf file, as it will usually be a standard A4 (which will result in very large blank space on top and bottom of the chart). Next approach will overcome this work-around.

Using Custom Size PDF Printing If desired, the use of a second program can be eliminated by doing some extra work on the original software used. The Figure 9 is obtained after such a process.
The algorithm followed in creation of the Excel chart in Figure 9 is as follows:

1. Select the chart in Excel and learn its original dimensions. For this chart, it is 20.04cm x 5.61cm.
2. Press Ctrl+P to show "Print Dialog" window.
3. Click "Page Setup", then "Page" tab, then "Options" button, then "Layout" tab, then "Advanced" button[5].

[5]Holy cracker, there should be a shortcut for this!

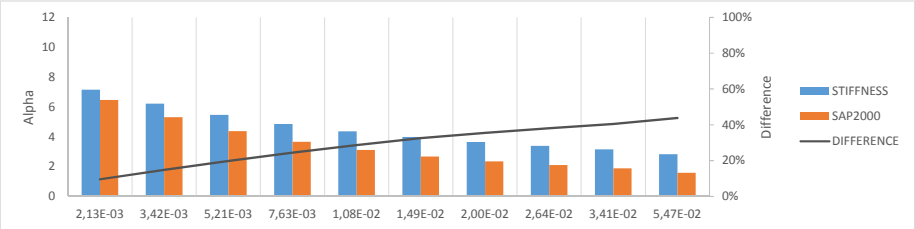


Figure 9: A Directly Obtained PDF Output

4. In paper size dropdown, select "PostScript Custom Page Size".
5. In the newly shown dialog, enter chart dimensions obtained in step 1. *Beware*, these are in mm whereas we learnt them as cm.
6. Close this dialog.
7. In the "Page Setup" dialog, set all the margins in the "Margins" tab to zero.
8. Print the graph as PDF.
9. Use the resultant PDF file as follows in LaTeX:

```
\begin{figure}[htbp]
\centering
\includegraphics[width=\columnwidth]{images/customPDF}
\caption{A Directly Obtained PDF Output}
\label{PDFDirect}
\end{figure}
```

To simplify things and make this approach a 1-click process, I tried to create an Excel add-on that will do steps 1 to 8 automatically. The thing is, it is possible to get dimensions of graph, convert it to cm (as it will be given in PostScript points), make margins zero, change printer to a pdf writer and print the chart. However, the adjustment stated in step 3 and 4 is strictly printer driver-specific. Therefore, it is practically not possible to change these settings in conventional ways (at least I could not find one); .NET native API does not provide support for specifying paper folding, stapling, paper size and other complex actions. In Windows, the DEVMODE data structure contains information about the initialization and environment of a printer or a display device, and it has a *dmDriverExtra* section which contains the number of bytes of private driver-data that follow this structure. It is assumed that the complex settings *live* here in an unknown format. The approach is to get memory dump of DEVMODE structure of a "configured" print driver, and before printing we can overwrite the spot of the DEVMODE on memory where this *private data* exists. Obviously it is a very 'unorthodox' work-around, and I did not implement this as it will be too much work for a rather simple task. Still, if anyone wants more info, it can be found on internet.

Another approach is to trim the white space in a PDF file automatically using either *Inkscape* or iTextSharp (*GhostScript* itself will work, too). I developed an Excel add-on SaveGraphAsPDF and a standalone app PDFTrimmer for this purpose. The second one uses Inkscape and the downside is warnings about group objects in PDF:

> PDF inclusion: multiple pdfs with page group included in a single page.

The reason is, PDF has a feature called 'Page Groups' (PDF Reference, section 11.4.7). These describe transparency effects between top-level objects on one page. When pdfTeX (or LuaTeX or XeTeX) includes a page from a PDF, it converts all pages into "Form XObjects" (section 8.10.1). pdfTeX also converts the Page Groups into /Group entries of the XObjects. The problem now is that Adobe products need also a /Group entry (whose content should not matter) in the /Page object which contains these XObjects to correctly render transparency (this is just needed to select the right rendering engine; the transparency information for the included pages should be taken from these included pages). pdfTeX will either use the first /Group it encounters when including PDFs or synthesize one when including PNGs with transparency. The warning is triggered when multiple Page Groups are encountered on one page (since the engine will then use the first one encountered and this may not be the "correct" one) and can probably be ignored. As a solution, PDFTrimmer recreates PDF by changing its color space to sRGB using GhostScript.

matplotlib matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell (ala MATLAB or Mathematica), web application servers, and six graphical user interface toolkits.

Generic Mapping Tools (GMT) Overview The Generic Mapping Tools, GMT, are an open source collection of tools for manipulating geographic and Cartesian data sets (including filtering, trend fitting, gridding, projecting, etc.) and producing PostScript illustrations ranging from simple xy plots via contour maps to artificially illuminated surfaces and 3D perspective views. It is released under the GNU Lesser General Public License.

5.2 Bibliography

A bibliography item is created within a `thebibliography` environment, and each entry starts with *bibitem[label]*{*marker*}. The *marker* is then used to cite the resource in within the document: *cite*{*marker*}.

If the *label* option is not used, the entries will get enumerated automatically. The parameter after the \`begin{thebibliography}` command defines how much space to reserve for the number of labels; {99} tells LaTeXto expect that none of the bibliography item numbers will be wider than the number 99.

For larger projects, BibTeXprogram is better suited. It allows to maintain a bibliographic database and then extract the references relevant to things cited in document.