Making the most of $\LaTeX$

Course notes

Tilo Arens

Dep. of Math. Sciences
Brunel University
Uxbridge UB8 3PH United Kingdom

email: `Tilo.Arens@brunel.ac.uk`

31/5/2000

Contents

1 Availability of LaTeX at Brunel University

If you want to use LaTeX at Brunel, there are a number of alternative installations you could choose to work with. The first question to ask is whether you want to work on one of the central workstations under Unix or under Windows 95/98/NT on one of the PCs.

1.1 Unix

Under Unix, one has the option of choosing one of a number of LaTeX installations. In general, working under Unix has the following advantages:

- the most complete installations are available, i.e. no additional software needs to be installed to make use of most features presented in this course.
- many additional tools are available, like drawing programs, software to manipulate post-script files,...
- the pre-viewer `xdvi` is available.

Arguably, the two biggest disadvantages are, that no standard graphical user interface and no spell-checker are available.

The standard installation provided by the computing services is activated by typing

```
use tex_v3.14
```

at the Unix prompt. This will provide the LaTeX 2ε version of 1/6/1994. Moreover, it does not include many of the standard additional packages like `babel` (multi-language support) or `seminar` (designing transparencies).

A better installation has been provided my Mike Warby. Activate it by typing

```
source /vol/pcpack/ma/staff/lab/mike/bin/tetex.st
```

at the Unix prompt. This will provide you with the LaTeX 2ε version of 1/12/1998. Moreover, there is hardly anything, I couldn't find in this installation. If you plan to use this package regularly, best append the above line to the file `.cshrc` in your home directory.

There are a few more installations available, about which I cannot say much. I definitely found the above two the most satisfactory ones so far.

In all cases, all the LaTeX commands can be activated from the Unix command-line. Any editor can be used to create and modify LaTeX documents.

1.2 Windows 95/98/NT

Up to my knowledge, Brunel University Computing Services does not provide any LaTeX package for use on networked Windows NT PCs. So this software has to be installed on individual PCs. The advantage of using the Windows is arguably that office-style graphical user interfaces and editors are available, which are very easy to use.

The nicest distribution of TeX currently available is MikTeX. It can be downloaded via the internet from the site

`www.miktex.de`

The installation is straight-forward, but should be carried out on a networked PC by the administrator (I can do this for you, or contact Kula).

A drawback of MikTeX is that not all advanced packages are available. But downloading additional packages from the web and installing them is very easy.

1.3 Editors that work with LaTeX

The input LaTeX accepts must be a plain simple text file. Thus, any text editor can be used in principal. However, a *good* editor will support the user by providing shortcuts for certain LaTeX commands and highlighting parts of the document to improve readability.

Many good editors also include buttons that will automatically execute the various LaTeX programs like `latex`, `bibtex`, `dvips` or a `dvi`-viewer.

1.3.1 Emacs and AucTeX

Emacs is the most powerful editor available for Unix. It is highly customizable, in fact it can be programmed to do almost anything. On the downside, Emacs is not straight-forward to use if you are unfamiliar with it. Once you have got to know it, you probably won't want to work without it. Today, there are two branches of Emacs: Gnu Emacs and Xemacs.

There are a number of extensions available in Emacs that allow things like automatic formatting, automatic insertion of commands and environments, starting of LaTeX programs by a mouse-click,... The best known of these is AucTeX. It is included in Xemacs, but not in Gnu Emacs, but it can be downloaded from the TeX archives at `www.tex.ac.uk`. AucTeX also includes a spell-checker.

1.3.2 WinEdt

`WinEdt` is a very nice editor and graphical user interface available under Windows 95/98/NT. It has short-cuts to start all TeX programs and includes a spell-checker. All LaTeX commands, labels and comments in your LaTeX document will be highlighted. `WinEdt` comes readily configured for use with MikTeX

However, different to TeX itself, `WinEdt` is shareware and not free. You can download the program to test it for 30 days from

`www.winedt.com`

An educational license for `WinEdt` version 5 costs 30 US dollars at the moment.

1.3.3 WinShell

`WinShell` is another editor and graphical user interface for TeX running under Windows 95/98/NT. It is very much like a little brother of `WinEdt`. It is not quite as user-friendly, provides less features and is more difficult to configure, but on the other hand it is free software. The current version of `WinShell` can be downloaded from

`www.winshell.de`

2 Help on LaTeX

Whenever you do not know how to do a certain task with LaTeX, it is time to look for help. Their are numerous ways to do this, the three most useful ones are

- buy/borrow a book
- ask a colleague
- look for online help

I will focus on the third option here.

Figure 1 indicates where documentation and software packages are located in your local TeX installation. The best introductory source on LaTeX that should be the part of any LaTeX distribution, is a file called `lshort.dvi`. In the Unix distribution, it is in the directory `texmf/doc/latex/general`. The full path, for Mike Warby's installation is

```
/vol/pcpack/ma/staff/lab/mike/latestmkwtetex/share
  /texmf/doc/latex/general/lshort.dvi
```

This file is missing from the MikTeX distribution, but it can be downloaded from

```
www.tex.ac.uk/tex-archive/info/lshort/english/
```

There are a lot more useful files in the `texmf/doc/latex` directory. Have a browse through the directory. For people not using LaTeX 2ε yet, I recommend the file `usrguide.dvi` in the `base` directory.

Another, very useful document, especially for information on things slightly out of the ordinary, is the frequently-asked-questions document of the German TeX users group. It is only available in German language, though. You can find it at

```
www.dante.de/faq/de-tex-faq/
```

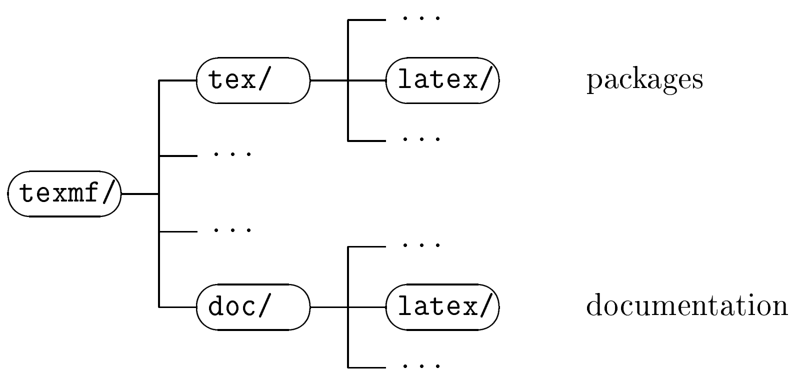


Figure 1: The LaTeX directory tree

3 Using BibTeX

The BibTeX-package is a powerful tool to automatically generate the bibliography for a LaTeX document. It relies on a database file containing the relevant information about the articles, books, theses, etc. The entries in the bibliography are formatted by rules laid down in a *style file*. Numerous such style files are already provided that format the entries according to the rules of certain journals (e.g. SIAM, AMS). But it is also possible to create a style containing individual preferences.

To use BibTeX, the following commands must be included in the LaTeX text:

```
\bibliographystyle{style}
\bibliography{database}
```

The first command defines a bibliography style which could, for example, reflect the style of a certain journal. The second command tells LaTeX which database to use for the bibliography. Instead of a single database, a whole list could be provided:

```
\bibliography{database1,databse2,...}
```

It is thus possible to have separate databases for e.g. books, journal articles specific to certain subjects, manuals, etc.

To make LaTeX generate the bibliography, the following sequence of commands is necessary:

```
latex myfile
bibtex myfile
latex myfile
latex myfile
```

The first `latex` command will produce lots of warnings because all the citations LaTeX cannot find an entry for. The `bibtex` command scans the database for the entries used in the text and generates the bibliography according to the selected style. The second `latex` command now includes the bibliography into its output, but the warnings will still appear. With the last command, all cross-references can finally be resolved — no more warnings should now appear.

3.1 The bibliography database

The bibliography databases used by BibTeX are normal text files, just like ordinary TeX files. However, they have the ending `.bib`, and their format is very different. Figure 2 is an example for a typical database file containing entries for a journal article, an unpublished article, a book and an article from a conference proceedings.

Other possible entries in a bibliography database include

`@phdthesis, @masterthesis, @manual, ...`

to name but few. A comprehensive list is given e.g. in [4,7]. Each of these entries requires certain fields to be defined while others are optional:

entry	required fields	optional fields
`@article`	`author` `title` `journal` `year`	`volume` `number` `pages` `...`
`@book`	`author` or `editor` `title` `publisher` `year`	`volume` `series` `edition` `...`
`@incollection`	`author` `title` `booktitle` `publisher` `year`	`editor` `volume` `series` `pages` `...`
`@manual`	`title`	`author` `...`
`@mastersthesis` `@phdthesis`	`author` `title` `school` `year`	`address` `month` `note` `type`
`@unpublished`	`author` `title` `note`	`month` `year`

All entries can also have the field `annote` that will be ignored by BibTeX and can be used to put comments into the database-entries, e.g. explaining something about the citation in question. A complete list of all possible fields is given in [4].

The first item in any database entry is always the label by which this entry will be referred to in a LaTeX document. Thus, the book by Gilbarg and Trudinger would be referred to using the command `\cite{gilbtrud83}`.

Depending on the style file you choose, a lot of reformatting will be done on the entries. To prevent errors the following guidelines should be observed:

- BibTeX uses the double quote `"` to delimit text fields. Thus if any double quotes appear in the field, these should be enclosed in curly brackets. The same is true for all umlauts (see Figure 2).
- Some styles change the initial letters of words in the title from upper to lower case. If this is to be prevented for certain words like names (e.g. the

word *Nyström*) in the example), either the whole word or the initial letter should be enclosed in curly brackets.

- The rules for author names are quite complicated. Normally one should start with the surname, then all first and middle names, separated from the first name by a comma. Individual author names should be separated by the word *and*. It is the a matter of choice whether full first names are given or just the initial. Some bibliography styles will automatically abbreviate all first and middle names.
- If in doubt, check the more detailed information given in [4, 7].

To make use of a database file *mybase.bib*, the LaTeX command

```
\bibliography{mybase}
```

would be used. LaTeX will then insert the bibliography at this place in the text.

3.2 The bibliography styles

The way in which the entries in the bibliography database are used in generating the bibliography is defined by the `\bibliographystyle` command. The styles are also stored in text files; they have the ending `.bst`. But the format of these files is quite complicated and the ordinary user will probably not want to involve him-/herself with this. A number of predefined styles are available, and it is also possible to define customized styles. The following styles are available:

plain Entries are sorted by author name, year and title. Each reference gets a number that is used in the `\cite` command.

unsrt Entries are sorted in the order in which they appear in the text. Otherwise like **plain**.

alpha Entries are sorted by author name, year and title. Referencing is done by an abbreviation of the author name and the year.

abbrv Like **plain**, but first names, month and journal names are abbreviated.

siam The bibliography style used by SIAM journals.

Figures 3–6 show how a bibliography using the database file in 2 would look like if each of these styles were used.

Figure 6 actually shows the use of a customized style file. To create a customized style file, simply type the command

```
latex makebst
```

One is now guided through a long list of options. At the end of this process, the style file is created automatically.

```
%% Example of a BibTeX database

@book{gilbtrud83,
  author    = "Gilbarg, D. and Trudinger, N. S.",
  title     = "Elliptic Partial Differential Equations of Second
               Order",
  edition   = "second",
  publisher = "Springer, Berlin",
  year      = 1983}

@article{jonesmpcps84,
  author    = "Jones, D. S.",
  title     = "An exterior problem in elastodynamics",
  journal   = "Math. Proc. Camb. Phil. Soc.",
  year      = 1984,
  volume    = 96,
  pages     = "173--182"}

@incollection{kirschipmp93,
  author    = "Kirsch, A.",
  title     = "Diffraction by periodic structures",
  booktitle = "Inverse Problems in Mathematical Physics",
  editor    = "P{\"a}v{\"a}rinta, L.  and Somersalo, E.",
  publisher = "Springer, Berlin",
  pages     = "87--102",
  year      = 1993}

@unpublished{meierjiea,
  author    = "Meier, A. and Arens, T. and Chandler-Wilde, S. N.
               and Kirsch, A.",
  title     = "A {N}ystr{\"o}m Method for a Class of Integral
               Equations on the Real Line with Applications to
               Scattering by Diffraction Gratings and Rough
               Surfaces",
  note      = "submitted for publication"}
```

Figure 2: A typical bibliography database

[1] D. Gilbarg and N. S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Springer, Berlin, second edition, 1983.

[2] D. S. Jones. An exterior problem in elastodynamics. *Math. Proc. Camb. Phil. Soc.*, 96:173–182, 1984.

[3] A. Kirsch. Diffraction by periodic structures. In L. Päivärinta and E. Somersalo, editors, *Inverse Problems in Mathematical Physics*, pages 87–102. Springer, Berlin, 1993.

[4] A. Meier, T. Arens, S. N. Chandler-Wilde, and A. Kirsch. A Nyström method for a class of integral equations on the real line with applications to scattering by diffraction gratings and rough surfaces. submitted for publication.

Figure 3: The bibliography using `\bibliographystyle{plain}`

[GT83] D. Gilbarg and N. S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Springer, Berlin, second edition, 1983.

[Jon84] D. S. Jones. An exterior problem in elastodynamics. *Math. Proc. Camb. Phil. Soc.*, 96:173–182, 1984.

[Kir93] A. Kirsch. Diffraction by periodic structures. In L. Päivärinta and E. Somersalo, editors, *Inverse Problems in Mathematical Physics*, pages 87–102. Springer, Berlin, 1993.

[MACWK] A. Meier, T. Arens, S. N. Chandler-Wilde, and A. Kirsch. A Nyström method for a class of integral equations on the real line with applications to scattering by diffraction gratings and rough surfaces. submitted for publication.

Figure 4: The bibliography using `\bibliographystyle{alpha}`

[1] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, Springer, Berlin, second ed., 1983.

[2] D. S. Jones, *An exterior problem in elastodynamics*, Math. Proc. Camb. Phil. Soc., 96 (1984), pp. 173–182.

[3] A. Kirsch, *Diffraction by periodic structures*, in Inverse Problems in Mathematical Physics, L. Päivärinta and E. Somersalo, eds., Springer, Berlin, 1993, pp. 87–102.

[4] A. Meier, T. Arens, S. N. Chandler-Wilde, and A. Kirsch, *A Nyström method for a class of integral equations on the real line with applications to scattering by diffraction gratings and rough surfaces.* submitted for publication.

Figure 5: The bibliography using `\bibliographystyle{siam}`

[1] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, Springer, Berlin, 1983, second edition.

[2] D. S. Jones, An exterior problem in elastodynamics, *Math. Proc. Camb. Phil. Soc.*, **96**, 173–182 (1984).

[3] A. Kirsch, Diffraction by periodic structures, in *Inverse Problems in Mathematical Physics* (L. Päivärinta and E. Somersalo, eds.), 87–102, Springer, Berlin, 1993.

[4] A. Meier, T. Arens, S. N. Chandler-Wilde and A. Kirsch, A Nyström method for a class of integral equations on the real line with applications to scattering by diffraction gratings and rough surfaces, submitted for publication.

Figure 6: The bibliography using a customized style

4 Presentations with `seminar.sty`

The `seminar` package provides a new document class for LaTeX. It allows the creation of slides with a minimum of additional work as compared to typing a normal LaTeX text. It is also very easy to produce notes for the audience by printing the slides in a reduced format, two per A4 page.

The basics of using the `seminar` document class

As a first example, let us create a simple slide with just one equation on it. The corresponding LaTeX file might look like this:

```
\documentclass[a4]{seminar}

\pagestyle{empty}

\begin{document}

\begin{slide}
Consider the equation:
\[ A x = b \]
\end{slide}

\end{document}
```

If you take a look at the resulting `.dvi` file, the result is a single page, in *landscape* orientation, with the text right in the middle. There is also a rectangular frame drawn around the slide. Note that the font-size has automatically been increased to a size which is appropriate for a transparency.

We can now start to do some slight modifications:

Portrait orientation: To create slides in portrait orientation, the tex file has to be modified in two places: The `\documentclass` command should be replace by

```
\documentclass[a4,portrait]{seminar}
```

and all `slide` environments should be replaced by `slide*` environments.

Changing the frame: The rectangular frame can be replaced by something else using the `\slideframe` command. The command `\slideframe{none}` corresponds to no frame at all, while `\slideframe{plain}` corresponds to the original rectangle. If `\usepackage{fancybox}` is used, the slide frame can also be drawn in the styles `double`, `shadow`, `oval` or `Oval`.

Font changing: Some people prefer to use a *sans-serif* font instead of the *roman* font used as default in LaTeX documents. To change the default font from *roman* to *sans-serif*, use the command

```
\renewcommand{\familydefault}{\sfdefault}
```

Unlike the use of `\sffamily`, this also changes the font in all `\section`, `\subsection`, ... commands.

2 slides a page: Simply add the command `\twoup` somewhere in the header of your document, i.e. before the `\begin{document}` command. This will reduce all slides by a factor of 50% and put two of them on each page.

More details on the options available with `seminar.sty` can be found in its documentation in the directory `texmf/doc/latex/seminar`. The file `sem-user.dvi` contains a detailed description.

4.1 Printing the slides

Basically, printing of slides works in exactly the same way as printing any ordinary LaTeX document: By using the program `dvips` the `.dvi`-file is converted to a post-script file which can then be printed either directly or by using `ghostview`. However, if the slides are in landscape format of if two portrait slides are to be printed on the same page using `\twoup`, `dvips` needs to no this. This is done by using the option landscape option:

```
dvips -t landscape -o myslides.ps myslides.dvi
```

4.2 Installation of `seminar.sty` for MikTeX

The `seminar` package is one of packages that are not included in the MikTeX distribution of LaTeX. But it is not difficult to install. The instructions given here in principal apply also to any other LaTeX package you wish to install.

1. Download the software from a CTAN server
   The nearest CTAN server is at Cambridge. You can access it via WWW at `www.tex.ac.uk` or via anonymous FTP at `ftp.tex.ac.uk`. The `seminar` package can be found under

   ```
   /tex-archive/macros/latex2e/contrib/other/seminar
   ```

   Create a directory

   ```
   texmf\tex\latex\seminar
   ```

   on your computer and download everything from the `input` directory on the server into it. Then create a directory

```
texmf\doc\latex\seminar
```

and download everything from the `doc` directory on the server into it. This completes the downloading process.

2. Update the MikTEX configuration
   On the Windows *Start Menu*, select `programs/miktex/maintenance`. Here you will see the option `Refresh filename database`. This will add the new package to MikTEX' database.

3. If required, run LATEX on the documentation files you downloaded.

This finishes the installation process.

5 Customizing LaTeX: commands, lengths and counters

Most elements of way in which LaTeX produces typeset pages can be influenced by the user. The three most important tools for this purpose are user-defined LaTeX-commands or environments, length variables and counter variables. Each of these will be dealt with here:

5.1 Defining your own commands

For many purposes, it is useful do define one's own customized LaTeX-commands. For example, I frequently use the mathematical operator "grad" which is not available as one of the standard mathematical functions. To define a command which produces this operator, the following line has to be included in the document head:

```
\newcommand{\GRAD}{\mathrm{grad}}
```

The first argument of the `\newcommand` macro is the name of the command, in this case `\GRAD`. The name of a command always has to start with a backslash (`\`). The second argument is the text that will be inserted, wherever this command is encountered in a LaTeX-document. Thus, the `\GRAD` command will simply produce the word 'grad' in normal roman print. Note, that it will only work in a mathematical environment!

Many commands in LaTeX also take arguments. To create such a command, a modified version of `\newcommand` has to be employed. Consider the following example:

```
\newcommand{\norm}[2]{\Vert #1 \Vert_{#2}}
```

Here, a command `\norm` is defined, taking two arguments. The arguments are used in the definition by writing `#` directly followed by the number of the argument. For example, typing

```
\[ \norm{u}{H^1(\bar{D})} \leq C \]
```

would produce the output

$$\Vert u \Vert_{H^1(\bar{D})} \leq C.$$

Any command in LaTeX can be redefined. Thus it is possible to replace existing commands with something else. To do this, simply use `\renewcommand` instead of `\newcommand`; their syntax is identical.

5.2 Environments

A so-called *environment* is a pair of `\begin` and `\end` commands with identical argument, like `\begin{equation}`, `\end{equation}`. These are defined using the `\newenvironment` command. An example would be

```
\newenvironment{italics}{ \itshape }{ \upshape }
```

The command `\begin{italics}` *would then switch to an italic font, while* `\end{italics}` would change back to the normal roman font.

Environments can also take arguments (as for example the `tabular` environment does). The definition is then similar to `\newcommand`: an optional parameter is then added, stating the number of arguments:

```
\newenvironment{the_name}[#args]{beg_def}{end_def}
```

5.3 Lengths variables

Lengths play an important role in arranging text, graphics, etc. on a page. LaTeX knows a number of different units to denote lengths:

`cm`	centimeters
`mm`	millimeters
`in`	inches
`pt`	points (1 in = 72.27 pt)
`pc`	picas (1 pc = 12 pt)
`em`	length of hyphen (—) in current font
`ex`	height of letter x in current font

So a length could be given as `4cm` or as `3ex`.

In addition to these units, LaTeX has the concept of length variables. A number of these are already predefined; examples are `\textheight` (the height of the text-body on one page), `\baselineskip` (the line separation) or `\parindent` (the amount of space by which the first line of a paragraph is indented). These can be used in the same way as units; consider the following example:

```
Hello \hspace{8\baselineskip} Hello
```

produces:

Hello   Hello

To define your own length variables, use the command

```
\newlength{\my_new_length}
```

which makes a new length variable called `\my_new_length` available.

The value of any length variable can be manipulated with the following commands:

```
\setlength{\my_new_length}{4cm}
\addtolength{\my_new_length}{2cm}
\settowidth{\my_new_length}{great}
```

The command `\settowidth` sets the length variable to the width of the text given as the second argument.

All these commands can also be used to manipulate LaTeX' built in length variables. For example, the command `\setlength{\parindent}{0cm}` would prevent the indentation of the first line in each paragraph.

5.4 Counters

Similarly to length variables, there are also integer number variables, called counters. One of the main differences between the two is that counter names do not start with a backslash (\). The following commands are available to manipulate counters:

```
\newcounter{myCounter}
\setcounter{myCounter}{4}
\addtocounter{myCounter}{2}
\stepcounter{myCounter}
\value{myCounter}
```

The command `\stepcounter` increases the value of the counter by one. The command `\value` simply returns the value of the given counter. It can be used, for example, in conjunction with `\addtocounter`, to add the values of two counters:

```
\newcounter{myCounter1}
\setcounter{myCounter1}{4}
\newcounter{myCounter2}
\setcounter{myCounter2}{3}
\addtocounter{myCounter1}{\value{\myCounter2}}
% \myCounter1 now has value 7
```

The value of a counter can be printed in a number of ways: as a number, a letter of even as one out of a list of symbols. Here are the output commands

`\arabic{counter}`	arabic number
`\Roman{counter}`	roman number, capitals
`\roman{counter}`	roman number, small letters
`\alph{counter}`	small letter
`\Alph{counter}`	capital letter
`\fnsymbol{counter}`	footnotesymbols

```
\begin{enumerate}
\renewcommand{\labelenumi}{%
   (\alph{enumi})}
\item the first item
\item the second item
\renewcommand{\labelenumi}{%
   (\arabic{enumi})}
\end{enumerate}
```

These commands produce:

(a) the first item

(b) the second item

Figure 7: Changing enumeration

As there are only 26 letters, using `\alph` or `\Alph` with a counter larger then 26 will result in an error. Similarily, there are only 9 different footnotesymbols.

Counters are used everywhere in LaTeX documents, for numbering sections, chapters, pages, references, equations, ... Figures 7 and 8 give some examples of how the layout of a document can be changed by simple counter manipulations.

```
% enumerate equations by section number in roman
% capitals followed by their number in the current
% section

\renewcommand{\theequation}{%
   \Roman{section}.\arabic{equation}}
```

Figure 8: Changing equation enumeration

6 LaTeX and Graphics

The basic story of LaTeX and graphics is simple: Postscript graphics are relatively easy to handle, everything else is more complicated. Together with the program `dvips`, LaTeX can handle postscript images. They can be scaled, rotated and combined with LaTeX typeset text. The best way to handle graphics in other formats is to convert them to postscript.

6.1 The `graphicx` package

The way to handle postscript graphics and related issues in LaTeX 2ε is to use the `graphicx` package:

```
\usepackage[dvips]{graphicx}
```

This command assumes, you are using `dvips` to translate from `dvi` to postscript. There are other possibilities, but they are hardly ever used.

An image, say one stored in a file called `some-pic.eps` can now be included at any place within a LaTeX document with the command

```
\includegraphics{some-pic.eps}
```

It is also possible to include ordinary `.ps` files, if they only contain one page. The file ending can be omitted, if the file is uniquely determined without it.

The `\includegraphics` command can also take a further optional argument to scale, rotate or clip an image. The optional argument is a list of keywords, each being assigned a value. An example could be:

```
\includegraphics[height=2cm,width=3cm]{some-pic.eps}
```

Some of the possible keywords are given in the following table:

keyword	possible values
width	length
height	length
scale	real value
angle	degress
origin	special
clip	*true* or *false*

The `angle` keyword is used for rotations. See Figure 9 for an illustration. These four pictures were included with the commands

```
\includegraphics{letterA}
\includegraphics[angle=90]{letterA}
\includegraphics[angle=180]{letterA}
\includegraphics[angle=225]{letterA}
```

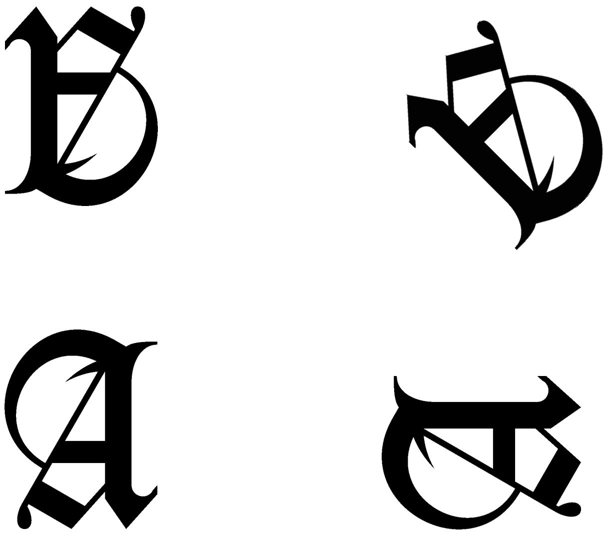


Figure 9: The letter A, rotated to four different angles

The rotation will normally be carried out around the point of reference for the image (normally the lower left corner). However, this can be changed using the `origin` key. It takes a combination of the arguments `lrctbB` as its values, these denoting *left*, *right*, *centre*, *top*, *bottom* and *baseline*. For more information, see the documentation file `texmf/doc/latex/graphics/grfguide.ps`.

6.2 Including other graphics formats

The standard way to include graphical images from any other format is to translate them to `eps` format and then to use the standard method to include them in your LaTeX document. Obviously, this has the disadvantage of increasing the size of most image files dramatically. On the other hand, there seems to be no feasible alternative.

Conversions of graphical images into postscript format may not be straightforward under Windows 95/98/NT, as postscript is not really embraced by Microsoft. Thus, it is strongly recommended to carry out this task under Unix. Here, it is not difficult at all:

1. Log onto any Brunel Unix workstation under *XWindows*. This means, either logon directly at the console of a Sun workstation or from any Windows

NT PC using *eXceed*.

2. At the Unix prompt, type

   ```
   use studentsoft
   xv &
   ```

   This starts the program `xv`, which is a shareware graphics conversion program.

3. After a few seconds, the `xv` window appears. Click the right mouse button on this window to open the `xv controls`.

4. Click on `load` (left mouse button) to load the graphics image. The program `xv` understands most graphics formats like `gif`, `tiff`, `jpeg`, `bmp` and a few others.

5. Click on `save` to save the image in postscript format. A dialog window will appear in which you can specify the filename and select the format (at the very top).

6. The `\includegraphics` command can now be used to include the image into a LaTeX document.

6.3 Related matters: Colour

Typesetting text in a colour different from black is intimately related to including graphics, as the same methods have been used by the developers of LaTeX to achieve both ends. The package to be used in the case of colour is `color.sty`. It is used by adding

```
\usepackage[dvips]{color}
```

to the document head. The optional parameter `dvips` can be omitted, if the `graphicx` package has been included previously.

There are number of different ways in which a colour is represented on a computer. The `color` package uses the following models:

named: This is the simplest model: 68 different colours have been defined already and can be used by specifying their names. You find these colours in the file

```
texmf/dvips/base/color.pro
```

rgb: The colour is specified as a combination of three numerical values, indicating how much red, green and blue hue, repsectively, should be used to make up the colour. The three values should be between 0 and 1.

cmyk: This is similar to `rgb`, but this time cyan, magenta, yellow and black hues are giving. This is the model used in most printers.

gray: One numerical value between 0 and 1 is given, specifying a grayscale colour.

To set text in a different colour, use the `\color` or the `\textcolor` commands:

```
\color[named]{ForestGreen}
Hi there
\color{black}
```

```
\textcolor[named]{ForestGreen}{Hi there}
```

These two examples will both produce the same result.

Notice in the first example above that no colour model has been specified when switching back to black printing. This is because LaTeX itself knows how the colour `black` is defined. Other pre-defined colours are *white, red, blue, green, cyan, magenta* and *yellow*. You can also define your own colours:

```
\definecolor{fgreen}{named}{ForestGreen}
\definecolor{my-purple}{rgb}{0.5,0.0,0.5}
\definecolor{dark}{gray}{0.2}
```

There are a few other commands available. The `\pagecolor` command specifies the background for a whole page. It has the same syntax as the `\color` command. Finally there are `\colorbox` and `\fcolorbox`, that produce colored boxes. As an example, consider

```
\colorbox[gray]{0.5}{This box is gray.}
\fcolorbox[named]{BrickRed}{ForestGreen}{Red frame,
     green background}
```

6.4 Related matter: rotation and scaling

The possibilities of scaling and rotation as outlined in the section on including postscript graphics can also be applied directly to LaTeX text. For this purpose, the `graphicx` package also includes the `\rotatebox` and `\scalebox` commands. Just have a look at the following examples:

```
\rotatebox{45}{rotated text}
```

rotated text

```
\scalebox{1.5}[2]{scaled text}
```

scaled text

The `\rotatebox` command takes as a first optional parameter a specification of the origin of the rotation as described for the `\includegraphics` command. Note that for `\scalebox`, the second parameter is optional. This parameter gives the stretch in the vertical direction and defaults to the stretch in horizontal direction (the first parameter) if it is omitted.

7 Diagrams in LaTeX documents

Technical documents often require diagrams to illustrate difficult arguments or constructions. There are a number of ways in which such diagrams can be created:

- use a drawing available on paper and glue a copy of this into the printed LaTeX document (not so long ago, this was the method of choice),
- scan a drawing available on paper and include it as a graphics file,
- use the LaTeX `picture` environment or PicTeX to create the diagram (there are other options here: TeXdraw,. . .)
- use an external program to produce the diagram and include it as a graphics file in postscript format.

Attention will be focused on the second two options here.

7.1 The `picture` environment

The `picture` environment is not necessarily the most sophisticated way to produce diagrams for LaTeX, but it can be useful in its way. Other, similar packages like PicTeX or TeXdraw certainly offer more possibilities. But for certain objectives, the `picture` environment can be just the right tool as you need nothing fancy except LaTeX to use it.

A typical use of the `picture` environment might look as in the following example:

```
\setlength{\unitlength}{1cm}
\begin{picture}(10,5)
\put(0,1){Some text}
\end{picture}
```

The first command here, `\setlength`, sets a length variable `\unitlength`. The value of this variable is the unit length for the coordinate system used to draw the picture. The second line starts the `picture` environment and defines the size of the picture: LaTeX will reserve a space of 10×5 units for the picture. As `\unitlength` is set to 1cm, this means 10cm by 5cm.

One of the nice features of the `picture` environment is that although LaTeX keeps exactly the specified space clear of everything else, nobody says that the actual diagram has to stay in that space (here the diagram is just a line of text). So the `picture` environment can be used to overlay text with graphics or something similar.

The `\put` can be used to put everything inside a diagram that can be part of a LaTeX document: text, mathematical formulae, boxes, postscript graphics,. . . In

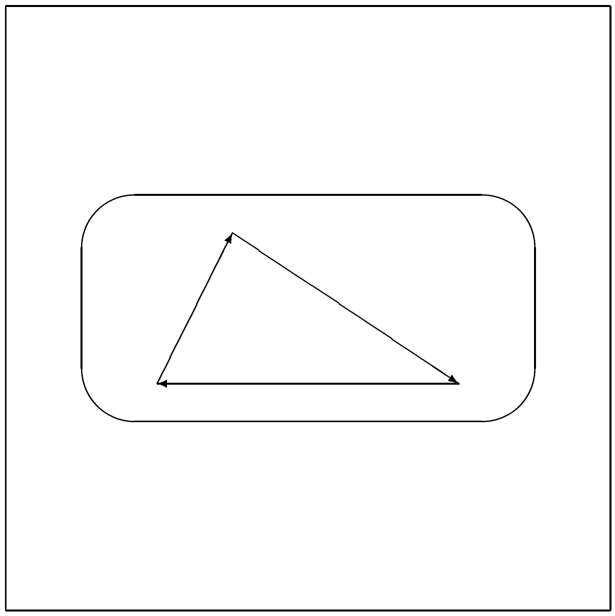


Figure 10: Example of a diagram obtained with the `picture` environment

addition, graphical objects like lines, arrows and rectangles with rounded corners (called ovals) can be put within a picture. An extension also makes Bezier splines possible. The origin of the coordinate system used by the `\put` command is the lower left-hand corner of the picture. Have a look at the example picture in Figure 10. It has been created by the following commands

```
\begin{picture}(8,8)
\put(0,0){\line(1,0){8}}
\put(8,0){\line(0,1){8}}
\put(0,8){\line(1,0){8}}
\put(0,0){\line(0,1){8}}
\put(4,4){\oval(6,3)}
\put(6,3){\vector(-1,0){4}}
\put(2,3){\vector(1,2){1}}
\put(3,5){\vector(3,-2){3}}
\end{picture}
```

For lines and vectors, the pair of numbers in the brackets behind the command indicate the slope of the line. This has to be a pair of whole numbers between -6 and 6 that have no common divisor. The `picture` environment's biggest limitation is that no arbitrary slopes for lines are allowed. For horizontal or vertical lines and vectors, the number in curly brackets at the end indicates the length of the respective object. For lines and vectors with any different slope, this number indicates *the length in horizontal direction*.

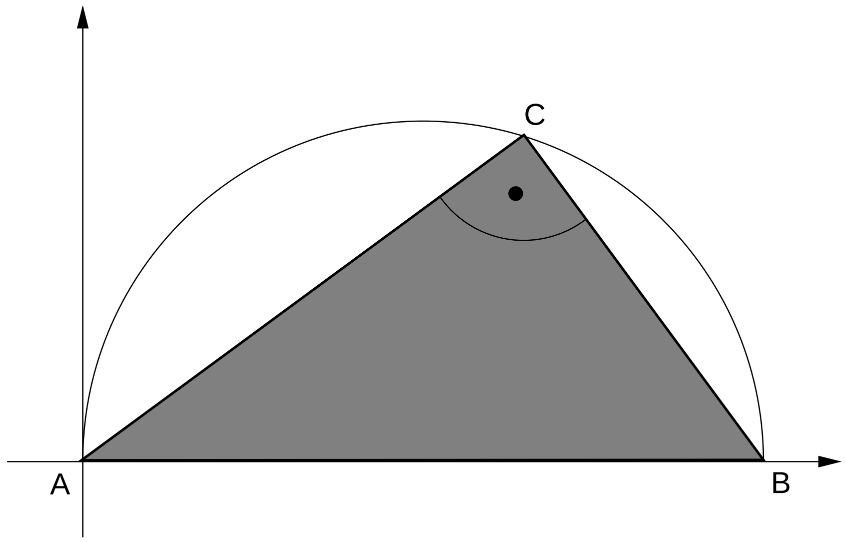


Figure 11: `xfig` demonstration: Theorem of Thales

For the `\oval` command, the coordinates in the `\put` command indicate the center of the oval. The pair of numbers in brackets behind the command itself indicate its dimension in horizontal and vertical direction, respectively.

Other commands like `\circle` or `\frame` work in a similar fashion. But producing diagrams of even limited complexity with the `picture` environment can be a complicated task and it is recommended to use special drawing programs for all but the simplest diagrams.

7.2 Using `xfig`

The Unix program `xfig` is an example of a program that can be used to create diagrams to include in LaTeX documents. It provides lines, arrows, ellipses, polygons, splines and many other graphical objects. Text can also be included, and a bonus is that `xfig` allows the use of either postscript or LaTeX fonts. Thus the fonts in an `xfig` drawing will look exactly those in LaTeX.

The finished drawing can be exported into one of several different graphic formats. For using `xfig` together with LaTeX, it is most advisable to export a drawing as encapsulated postscript (`.eps`). Figure 11 shows an example of a diagram created with `xfig` and included in this way.

7.3 Matlab Graphics and LaTeX

Another program often used to create figures that are to be included in articles or other reports is `Matlab`. Principally, it is very easy to do this: Using the

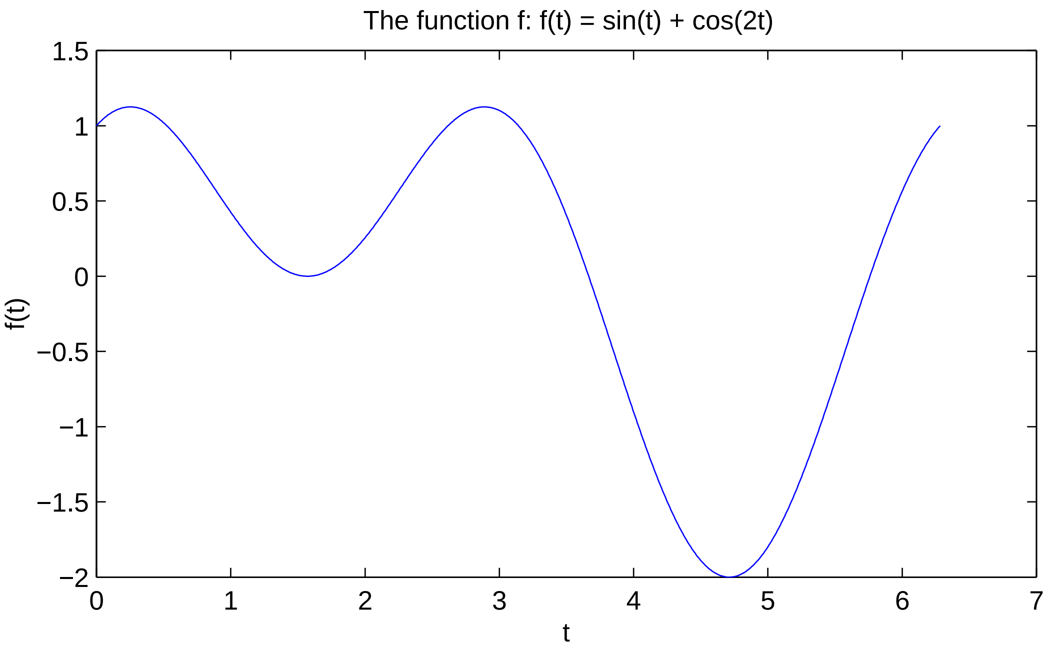


Figure 12: First version of a `Matlab` figure

`Matlab` `print` command with the option `-deps` (or `-depsc` for colour diagrams), the figure can be saved as a `.eps` file.

The difficulties arise in the detail, when trying to make the figure look good. Figure 12 shows the graph of a function plotted in Matlab and then included using `\includegraphics`. It looks fine, but the fonts do not match the ones used in LaTeX. A better result, as in Figure 13, can be achieved by changing the fonts `Matlab` uses. This is done by typing

```
set(gca,'FontName','Times-Roman','FontSize',12)
```

in `Matlab` before adding any titles or axis labels to the graph. The result is still not fully satisfying, however, as `Matlab` uses postscript fonts and LaTeX normally does not. Even better results could be achieved by telling LaTeX to use postscript fonts as well (see the next section).

There is a second possibility, which is also very useful if it is intended to use the scaling options of the `graphicx` package: One could combine the `picture` environment of LaTeX with `Matlab` graphics. Tell `Matlab` not use any axis by typing

```
axis off
```

Then all axis information and labels can be placed by LaTeX using the `\put` command. Of course, the `\includegraphics` command must also be used within a `\put` command.

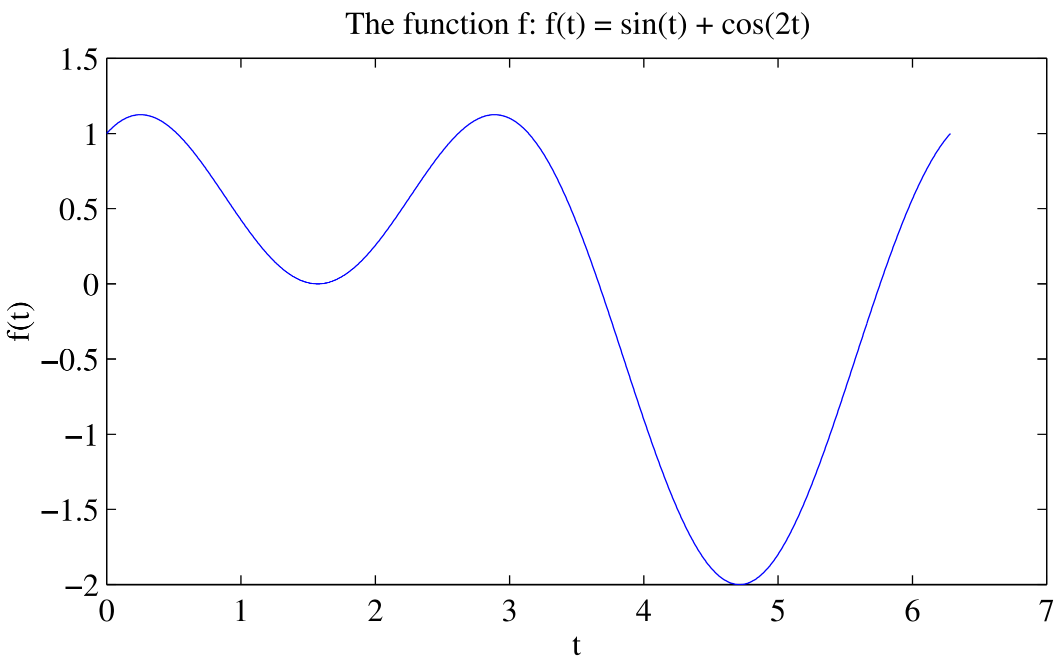
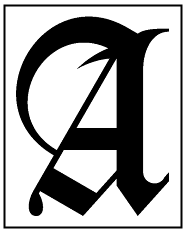


Figure 13: Second version of a `Matlab` figure

7.4 Text flowing around floats



There are several possibilities to accomplish the effect of a figure, picture or drawing with the normal text *flowing* around it. One of the most useful ones is the package `wrapfig`. This package simply provides the `wrapfigure` environment. The picture can then be placed inside this environment. The example of the letter A in this paragraph has been included with the following commands:

```
\begin{wrapfigure}{r}{0.2\textwidth}
\fbox{\includegraphics{letterA.eps}}
\end{wrapfigure}
```

In this example, the `wrapfigure` environment takes two arguments: The first one indicates the position of the wrapped figure. Other possibilities are `l` (left), `i` (inside) and `o` (outside). The second argument is the width of the space to be reserved for the figure. The height is normally calculated automatically, but can also be specified explicitly by as *a number of lines* in an optional argument which would go before the position argument. There is also a `wraptable` environment which is similar to `table`. For more details see the file `wrapfig.sty` file which can be found in

```
texmf/tex/latex/misc
```

The `wrapfig` package is not part of the standard `MikTeX` distribution , but it can be downloaded from `ftp.tex.ac.uk` under the path

```
tex-archive/macros/latex/contrib/other/misc
```

Let me finish with a warning: This way of including a figure is not entirely natural to LaTeX and thus there can be situations in which the wanted result is not achieved by `wrapfig`. In this case, some special creative cheating is necessary.

8 Using Fonts and Layout Options

One of the unfortunate side-effects of using LaTeX is that all documents look the same. There seem to be very little possibilities for variation. But actually, there are quite a few options available to the user, they are just not documented in most of the introductory literature. The main two possibilities are using different fonts and changing the page layout.

8.1 LaTeX and fonts: metafont

It is possible to change fonts or use different fonts for part of a document with relatively simple changes to the document. This is made possible by the way LaTeX handles fonts together with a program called `metafont`. This program can create fonts of any size from special input files ending on `.mf`.

When typesetting pages, LaTeX itself is not interested in how the letters it typesets actually look. It just uses the sizes of the letters together with certain other geometrical constants to ascertain where every letter should be placed. For every font used by LaTeX, this information is contained in a `.tfm` (TeX font metric). It is interesting to note that this file is independent of the actual size of the font; this metric information can be scaled arbitrarily.

Only when the `.dvi` file is interpreted by a dvi-previewer or a printer driver, are the letters of each font used in the individual sizes. If they do not yet exist, they are then created by the program `metafont`. Simply speaking, each letter is described by a series of brush strokes with a brush held in a certain position and following a certain line. Each letter is drawn in the correct size and the resulting bitmap saved in a file with the ending `.pk`.

8.2 Activating a new font

Each font in LaTeX has five attributes: encoding, family, series, shape and size. If a new font is to be used, the first four of these can be set using the `\usefont` command. It is not normally necessary to invoke this command directly; more details on this can be found in the file

```
texmf/doc/latex/base/fntguide
```

Different fonts are usually provided through a LaTeX package that can be activated through the `\usepackage` command. The corresponding `\usefont` commands are implemented in these packages and the user need not be concerned about them. Some packages declare a font as default font for an entire document, while others provide commands to switch to certain fonts within a document. There is also a general distinction in TeX between text fonts and mathematical fonts.

Giving a complete picture of the issue of fonts is beyond this course. The following table gives some examples of text fonts and the names of the corresponding packages:

font name	package	commands
Concrete	`concrete`	provides default
Pandora	`pandora`	provides default
Gothic	`oldgerm`	`\gothfamily`, `\textgoth{...}`
Schwabach	`oldgerm`	`\swabfamily`, `\textswab{...}`
Fraktur	`oldgerm`	`\frakfamily`, `\textfrak{...}`

On the other hand there are mathematical fonts, i.e. fonts that can only be used in mathematical mode. The package `concrete`, for example, also changes the mathematicl font (if you only want to change the text font, use the package `ccfonts` instead). The American Mathematical Society (AMS) has provided some such mathematical fonts, also providing a great number of additional symbols.

The AMS fonts can be activated by adding

```
\usepackage{amssymb}
```

to the document head. First of all, this makes the blackboard letters, i.e. the symbols for natural, integer, rational, real and complex numbers available:

```
\[ \mathbb{N \qquad Z \qquad Q \qquad R \qquad C} \]
```

results in

$$\mathbb{N} \qquad \mathbb{Z} \qquad \mathbb{Q} \qquad \mathbb{R} \qquad \mathbb{C}.$$

The `amssymb` package also provides a great number of additional mathematical symbols. A list of these can be found in the file

```
texmf/doc/fonts/amsfonts/amsfndoc.dvi
```

8.3 Using Postscript fonts

Postscript fonts are special in that the fonts themselves are already provided ny the company `Adobe` and do not need to be created by `metafont`. On the other hand, these are the fonts normally used by many Windows applications (accessed as *true type fonts*). Thus, you may wish to use postscript fonts to make a LaTeX document look like an MS-Word document, for example (who would wish that? Well, it has happened...)

In LaTeX, there are a number of packages that activate one or more of the postscript fonts. This means that they change one or more of the three LaTeX commands `\rmdefault`, `\sfdefault` or `\ttdefault` which activate normal, sans serif or typewriter style fonts, respectively. Here is the list of which fonts are activated by which packages.

package	\rmdefault	\sldefault	\ttdefault
avant		Avant Garde	
bookman	Bookman	Avant Garde	Courier
helvet		Helvetica	
newcent	New Century Schoolbook	Avant Garde	Courier
palatino	Palatino	Helvetica	Courier
times	Times Roman	Helvetica	Courier

When using the package `times`, it is advisable also to use the additional package `mathptm` which activates corresponding fonts for the mathematical mode.

8.4 Choosing a standard font

Usually, LaTeX always uses the roman (normal shape) font for typesetting text. All other fonts must be activated separately. In LaTeX 2ε, the following font changing commands are used:

```
\rmfamily    \mdseries    \upshape
\sffamily    \bfseries    \itshape
\ttfamily                 \slshape
                          \scshape
```

There are corresponding commands for changing the font for a bit of text passed to the command as an argument:

```
\textrm{...}    \textmd{...}    \textup{...}
\textsf{...}    \textbf{...}    \textit{...}
\texttt{...}                    \textsl{...}
\emph{...}                      \textsc{...}
```

So the two command sequences

```
\bfseries bold text \mdseries
```

and

```
\textbf{bold text}
```

are equivalent.

However, these font changing commands do not effect text that LaTeX sets in different fonts by default, especially the fonts used for chapter and section headings. To change the default font throughout a document, including all headings, the following command, or a similar one, has to be included in the head of the document:

```
\renewcommand{\familydefault}{\sfdefault}
```

As indicated in the last section, the three commands `\rmdefault`, `\sfdefault` and `\ttdefault` each select a standard font with normal, sans serif or typewriter style. This example above would set the default document font to the default sans serif font thus changing the font style from roman to sans serif throughout the document.

8.5 The LaTeX page layout

Figure 14 demonstrates the way in which LaTeX arranges text on a page. It also shows the values of the appropriate length variables for the current document. You can get this graph by using the package `layout`. The command `\layout` will then produce this layout diagram with the corresponding values. Changing the values of the length variables will result in a different layout and can thus be used to change the way a LaTeX document looks.

There are also a number of packages available for typesetting text for different paper formats. These can normally be found in the directory

```
texmf/tex/latex/misc
```

Formats available include different styles for A4 and A5 size paper. If they are not yet installed, they can be found on the CTAN archive.

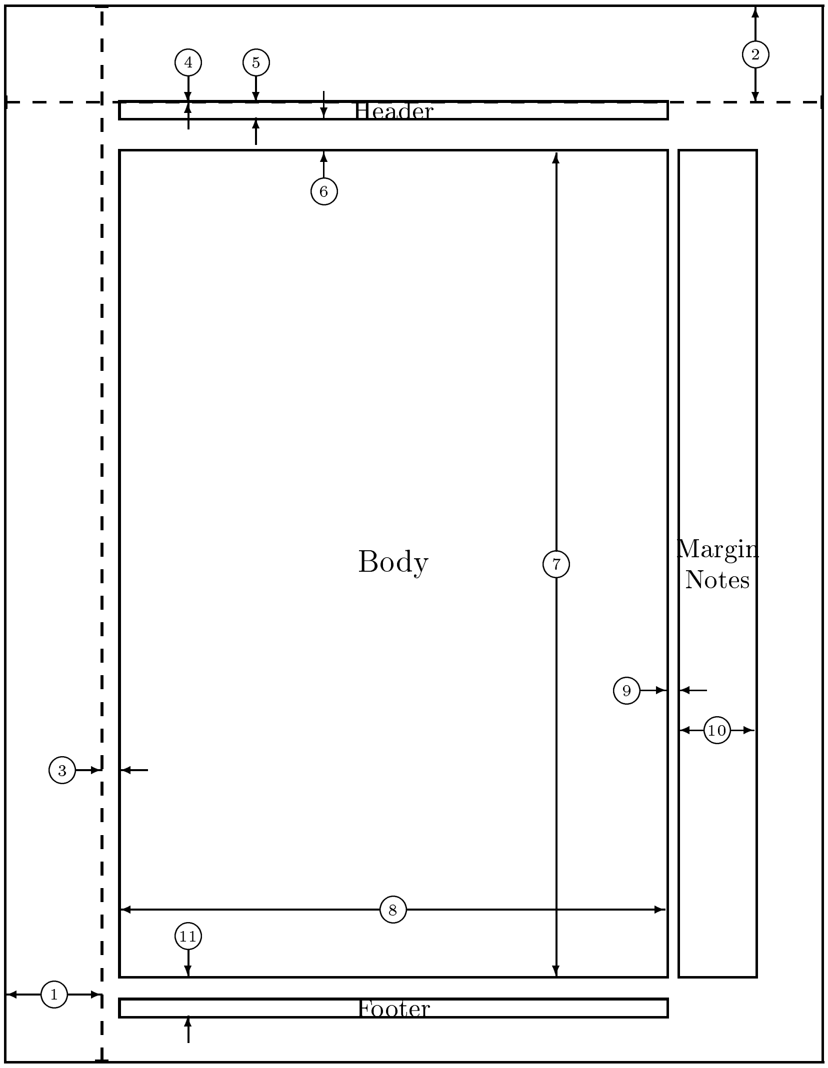
Other options include typesetting text in several columns using the `multicol` package. This package makes the `multicols` environment available. For example, the current paragraph is created using the commands

```
\begin{multicols}{2}
...
\end{multicols}
```

An additional heading can be added above the columns by typing

```
\begin{multicols}{2}[Heading]
```

You can switch back and forth between normal text and text typeset in columns any number of times in a document.



1	one inch + \hoffset	2	one inch + \voffset
3	\oddsidemargin = 14pt	4	\topmargin = 0pt
5	\headheight = 12pt	6	\headsep = 25pt
7	\textheight = 621pt	8	\textwidth = 411pt
9	\marginparsep = 10pt	10	\marginparwidth = 57pt
11	\footskip = 30pt		\marginparpush = 7pt (not shown)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 614pt		\paperheight = 794pt

Figure 14: The LaTeX page layout

9 Tips and tricks

9.1 The `\afterpage` command: managing floats

The way in which LaTeX handles the positioning of so-called *floats* (i.e. `figure` and `table` environments) can be a nuisance. Loosely speaking, floats are positioned at or near the point where they are inserted in the LaTeX document if enough space is available. Unfortunately, LaTeX often has quite a different idea of what *enough* space is than most users. If there is not enough space available, the float (and all floats following it) are collected and placed on pages immediately following the next `\clearpage` command. This command can be part of a `\chapter` or similar structuring command and thus not immediately obvious.

It follows that one way to force LaTeX to include all collected floats at some point in the text is to insert a `\clearpage` command. Unfortunately, this strategy makes later changes to the paper size or the insertion of text at an earlier point impossible without the risk of ruining the overall layout.

The `afterpage` package may be used to avoid this. It just provides one single command `\afterpage`. This command takes a sole argument, and this argument will be the next thing LaTeX processes *after finishing the typesetting of the current page*. Thus, the command

```
\afterpage{\clearpage}
```

will result in all remaining floats being included after the page with this command is finished.

9.2 Long tables

One of the disadvantages of the `tabular` environment is that the table it creates has to fit on one single page. Longer tables have to be separated manually into a number of parts that use the space optimally.

An alternative is the `longtable` environment provided in the `longtable` package. With this environment, tables can be created that extend over several pages. An initial head and final footer can be defined as well as intermediate headers and footers at the beginning and end of intermediate pages, respectively. The whole structure takes the following form:

```
\begin{longtable}
   % here goes the text for the initial header
\endfirsthead
   % here goes the text for intermediate headers
\endhead
   % here goes the text for intermediate footers
\endfoot
```

```
   % here goes the text for the final footer
\endlastfoot
   % now follows the text for the table rows,
   % like in the tabular environment
\end{longtable}
```

Because of the way such an environment is processed, it can sometimes take several runs of LaTeX before a `longtable` is typeset correctly. In most situations, 3 runs of LaTeX should be enough.

10 LaTeX and the Web

The World Wide Web is more and more becoming one of the most important ways of communicating in the modern world. Home pages are seen as a mixture of a calling card and a CV. Thus it is of great interest, to be able to publish papers, abstracts, but also student notes or exam papers on this medium. Remote teaching is also a possibility. Thus the question arises of how to get mathematics and thus LaTeX onto the Web.

The best way to get mathematics onto the Web is probably `MathML`, an `XML` mark-up language for describing mathematical formulae. Sadly, this language is not yet supported by today's browsers. Thus, one has to stick to `HTML`.

10.1 TeX-documents for downloading

The easiest way to make LaTeX documents available via the web is to simply create an `FTP`-link to them from an `HTML` page. When clicking onto the link, the file is downloaded onto the reader's computer, and maybe even displayed immediately, if the setup is correct. The question now is, which format to use:

DVI: The disadvantage of making the `.dvi` file available, is that this requires the reader to have a LaTeX previewer and all the correct fonts. Also, all images that are used in the file will have to be downloaded separately.

PS: Postscript is a good way to make a file available. Especially, all images would be part of the postscript file. The disadvantage is, that Windows users would require `GSview32` or a similar program to read and print the file.

PDF: A `.pdf` (portable document format) file is maybe the best options, as this format is used extensively and most users have some kind of `pdf`-viewer installed on their system (`acrobat reader`, `acroread`). The question is how to best create the `.pdf` file.

It is probably best, to make an article available in both `.pdf` and `.ps` format, thus providing two formats one of which should be readable on any machine.

A `.pdf` file can be created from a LaTeX document in one of three ways:

- Use `dvips` to create a postscript file and then convert it to pdf using `ps2pdf` (on Unix systems). This option is really easy, but the resolution of the resulting `.pdf` file does not seem to be to brilliant in the examples I tried.
- Use the program `dvipdfm` to directly convert the `.dvi` file to `.pdf`. This program should be part of most TeX installations today. It is quite a nice option as long as you do not have any graphics in the LaTeX document. The program `dvipdfm` cannot include postscript images. (Maybe it works

to convert them to pdf and then to include them...I have not tried that yet).

- Use `pdflatex` to directly create a `.pdf` file from a LaTeX source. This method probably produces the best `pdf` code, and special commands can be included in your LaTeX source file to add special `pdf` features like links and clickable cross-references. The disadvantage in some peoples eyes may be, that you are not using TeX itself any more. Also, postscript images cannot be included, but it works to convert them to `pdf` format first, as described below.

There remains the problem to include postscript graphics. Just two steps are necessary:

1. Store `pdf` versions of all `eps` files to be included. This can be done with the program `epstopdf` on a Unix system.

2. Change the graphics driver for the `graphicx` package:

```
\usepackage[pdftex]graphicx
```

After this, the modified LaTeX source can be used to create a `.pdf` file with `pdflatex`.

10.2 tth

The program `tth` is a very simple LaTeX to `HTML` converter available on Unix systems, but it can be compiled on other systems as well. It creates an `HTML` file solely consisting of text. Even mathematical formulae are typeset using normal text characters. Cross-references and citations are displayed as links to the corresponding place in the text. The program call would look like

```
tth -Lexample < example.tex > example.html
```

The `-L` option is used to tell `tth` what the name of the auxiliary file (`.aux`, `.bbl`, `.toc`,...) are. This important for including the cross-references.

10.3 latex2html

The program `latex2html` is a more sophisticated type of converter. At Brunel University, it is available as part of the Unix `tex_v3.14` software package. It converts mathematical formulae into graphical images which are loaded into the `HTML` file. The document is also spread out over several `HTML`-pages, one for each section or subsection. The degree of this spreading can be chosen by the user, as well as a lot of other features. Again, cross-references and citations are shown as links. The program is called by the command

```
latex2html example
```

A final warning

If you intend to convert from LaTeX to `HTML`, bear this in mind when creating the LaTeX document. Especially, avoid all mathematics inside text lines, all formulae should be displayed. Otherwise, you will probably not be satisfied with the result.

References

[1] M. Goosens, F. Mittelbach and A. Samarin, *The LaTeX Companion.*, Addison-Wesley, Reading MA, 1994.

[2] M. Goosens, S. Rahtz and F. Mittelbach, *The LaTeX Graphics Companion.*, Addison-Wesley, Reading MA, 1997.

[3] D. E. Knuth, *The TeXBook*, volume A of *Computers & Typesetting*, Addison-Wesley, Reading MA, 1986.

[4] H. Kopka, *LaTeX Einführung — Band 1*, Addison-Wesley, 1994, second edition.

[5] H. Kopka, *LaTeX Ergänzungen — Band 2*, Addison-Wesley, 1997, second edition.

[6] H. Kopka and P. W. Daly, *Guide to LaTeX2ε, Document Preparation for Beginners and Advanced Users.*, Addison-Wesley, Reading MA, 1999, 3rd edition.

[7] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading MA, 1994, second edition.