Practical scientific writing

Кулябов Д. С. Королькова А. В. Геворкян М. Н.

Table of contents

Preface

Practical scientific writing course.

Summary

In summary, this book has no content whatsoever.

Introduction

This course explains the basics of what LaTeX is and how it works in contrast to common word processors such as Microsoft Word or LibreOffice Writer.

Unlike common word processors such as Microsoft Word or LibreOffice Writer, LaTeX usually does not provide WYSIWYG ('What You See Is What You Get'). With LaTeX one takes plain text and enriches it with markup. This markup tells LaTeX about the logical meaning of certain elements of the text, similar to the way HTML does.

Take for example the element `<h2>` indicating a new section in an HTML document. LaTeX also has a command for this; here one would use the `\section` command.

The LaTeX workflow

Because LaTeX files are not the document itself but rather instructions on what each part of the document should be, you don't normally give other people your LaTeX file itself. Instead, after writing your LaTeX *source*, you run LaTeX on the file (normally using a program called `pdflatex`) to create a PDF file.

Multiple LaTeX runs

For simple files, you only need to typeset your file once to get the completed PDF. But once you start adding more complicated things, like cross-references, citations, figures, and tables of contents, you might need to run LaTeX more than once.

LaTeX campanion

LaTeX is not a single program. We will look at some other programs, and why you might want to use them, later in the course.

Formats and engines

For most of our examples, we don't use a program called `latex` but instead one called `pdflatex`. This is one of a family of related programs, all of which are 'descendants' of `latex`.

LaTeX is built on a system called TeX. We call LaTeX a 'format': a collection of macros (instructions and commands) that TeX understands. When you run `pdflatex`, you are *actually* starting a program called 'pdfTeX' with a pre-loaded 'LaTeX format'. We normally call pdfTeX an *engine*: a program that understands TeX instructions.

There are three engines in common use today:

- `pdfTeX`
- `XeTeX`
- `LuaTeX`

These are specialized engines for vertical typesetting. LuaTeX can also do a lot of this, but at the moment upTeX, in particular, is still the most popular system for Japanese.

1

LaTeX basics

1.1 Installing TeXlive

Installing the distribution *TeXlive*.

1.1.1 General information

- TeX Live — the most complete LaTeX distribution supported by the TeX community.
- Supports a large number of operating systems.
- Developed since 1996.
- Based on the teTeX distribution.
- MacTeX — a variant for MacOS.
- Main page: https://www.tug.org/texlive/.
- TeX Live — is a distribution with continuous updates as part of the annual version of the distribution.

1.1.2 Installation from distribution packages

- Ubuntu:

```
apt install texlive-full
```

- Windows. Use the Chocolatey package manager.

```
choco install texlive
```

1.1.3 Manual installation

- Links on the site are to mirrors. The mirror is selected automatically.
- Download the installer:
- Unix: https://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz

```
cd /tmp/
wget https://mirror.ctan.org/systems/texlive/tlnet/install-tl-
 ↪  unx.tar.gz
```

- Windows: https://mirror.ctan.org/systems/texlive/tlnet/install-tl-windows.exe
- For Windows: run the executable file and install.
- For Linux
- Unpack the downloaded file:

```
tar xzvf install-tl-unx.tar.gz
```

- Go to the unpacked directory and run the installer:

```
cd install-tl-[0-9]*
sudo ./install-tl
```

- It is recommended to create links to executable files in the `/usr/local/bin` directory. To do this, in the console version of the utility, select the 0 option, and then L. To return to the previous menu, use R.

1.1.4 Updating to the next version of TeXlive

- It is recommended to install the new version of TeXlive separately.
- But you can do a manual update using an existing installation.
- Let's assume that our architecture is `x86_64-linux`.
- If you have installed symbolic links to system directories (via installer option or `tlmgr path add`), remove them:

```
tlmgr path remove
```

- Move the entire TeXlive directory to match the new version, for example:

```
mv /usr/local/texlive/2024/ /usr/local/texlive/2025
```

- Remove package backups:

```
rm /usr/local/texlive/2025/tlpkg/backups/*
```

- Create links to executables:

```
/usr/local/texlive/2025/bin/x86_64-linux/tlmgr path add
```

- Download the latest version of the script `update-tlmgr-latest.sh`:

```
wget https://mirror.ctan.org/systems/texlive/tlnet/update-tlmg⌋
↪ r-latest.sh -O /tmp/update-tlmgr-latest.sh
```

- Run the script:

```
sh /tmp/update-tlmgr-latest.sh -- upgrade
```

- If you do not want to use the default repository for downloading new files, then replace it:

```
tlmgr option repository <reponame>
```

- Update the TeXlive package manager:

```
tlmgr update --self
```

- Update TeXlive packages:

```
tlmgr update --all
```

- Set symbolic links to executables in system directories (`/usr/local/bin`):

```
tlmgr path add
```

- You can recreate the cache *lualatex* under the user:

```
mv ~/.texlive2024 ~/.texlive2025
luaotfload-tool -fu
```

- If you don't do this, the cache will be recreated on the first run of `lualatex`.

2

LaTeX document structure

2.1 LaTeX document structure

This lesson shows the basic structure of a LaTeX document, and how to build it into a PDF file, as well as the main special characters used to control LaTeX.

Your first LaTeX document is going to be very simple: the idea is to show you how a document looks and how to typeset it successfully.

If you are using a local LaTeX installation, in your editor create a new file called `first.tex`, and either copy–paste the text below or type it in.

```
\documentclass{article}
\usepackage[T1]{fontenc}

\begin{document}
Hey world!

This is a first document.
\end{document}
```

Save the file and run:

```
pdflatex first.tex
```

You should get a PDF file that contains the text above *plus* a page number; LaTeX adds that automatically.

View the output `first.pdf` with whatever program you prefer for PDF viewing. Looks great; congratulations!

Errors happen. Check that you have entered each line in the text file exactly as written above. Sometimes seemingly small input changes give large changes in the result, including causing a document to not work. If you are stuck, try erasing the document and copying it fresh from the lines above.

If your LaTeX typesetting run ends with a question mark then you can get out by typing `x` and `Enter`.

LaTeX's error messages try to be helpful, but they are not the same as messages in word processors. Some editors also make it hard to see the 'full' text of an error, which can hide key details. LaTeX always creates a log of what it is doing; this is a text file ending in `.log`. You can always see the full error messages there, and if you have a problem, expert LaTeX users will often ask for a copy of your log file.

2.1.1 What you've got

The first document shows the basics. LaTeX documents are a mixture of text and commands. The commands start with a backslash and sometimes have arguments in curly braces (or sometimes optional arguments in square brackets). Then you get an output PDF by telling LaTeX to typeset your file.

Every LaTeX document has a `\begin{document}` and a matching `\end{document}`. Between these two is the *document body*, where your content goes. Here the body has two paragraphs (in LaTeX you separate paragraphs with one or more blank lines). Before `\begin{document}` is the *document preamble*, which has code to set up the document layout. The `\usepackage` command is used in most examples on this site to set up the font encoding.

LaTeX has other `\begin{...}` and `\end{...}` pairs; these are called *environments*. You must match them so that for every `\begin{x}` there has to be an `\end{x}`. If you nest them, then you must have `\end{y} ... \end{x}` to match `\begin{x} ... \begin{y}`, i.e. the `\begin` and `\end` statements matching in order.

We can add comments to a LaTeX file by starting them with %; let's use that to show the structure:

```
%% The document class with options
\documentclass[a4paper,12pt]{article}
%% Select T1 font encoding: suitable for Western European
 ↪  Latin scripts
\usepackage[T1]{fontenc}
%% A comment in the preamble
\begin{document}
%% This is a comment
This is   a simple document\footnote{with a footnote}.

This is a new paragraph.
\end{document}
```

You can see above that we've got two paragraphs: notice the use of a blank line to do that. Also notice that multiple spaces are treated as a single space.

You might also sometimes want a 'hard' space that does not break over lines: in LaTeX we can create that using ~, 'tying' two pieces of text together. That's particularly useful when we start creating cross-references later in the course.

2.1.2 Running LaTeX

LaTeX documents are simply plain text. To see this, try opening your first document in a simple text editor. You should see the same text as in a dedicated LaTeX editor, but without any highlight of keywords.

You can also convert to PDF without your editor; this means using the Command Prompt/Terminal, so don't worry if you are not familiar with this. If you *are*, you can navigate to the directory containing your `.tex` source file and run

```
pdflatex first
```

or

```
pdflatex first.tex
```

to typeset your PDF. Notice that the `.tex` extension is optional: LaTeX will assume files end with `.tex` unless you specify otherwise.

2.1.3 Special characters

You've probably spotted that \, { and } have a special meaning to LaTeX. A \ starts an instruction to LaTeX: a 'command'. The curly brace characters { and } are used to show *mandatory arguments*: information that commands require.

There are some other characters with special meaning; we've just seen that ~ is a 'hard' space, for example. Almost all of these characters are *very* uncommon in normal text, which is why they were chosen for special meanings. If you do need to show one of these special characters, we've put some information in the details table.

If you need to type in a special character, most of the time you can simply use a backslash in front of it, so for example \{ is used to print a literal {. There are a few cases where you need to use a longer command instead:

Symbol	Short Command (math and text)	Long Command (for text only)
{	\{	\textbraceleft
}	\}	\textbraceright
$	\$	\textdollar
%	\%	
&	\&	
#	\#	
_	_	\textunderscore
\		\textbackslash
^		\textasciicircum
~		\textasciitilde

For the last three symbols there are no short commands available, because \\ is used to indicate a linebreak and \~ and \^ are used to produce tilde and circumflex accents when using only ASCII characters as input.

2.1.4 Exercise

Try adding text to your first document, typesetting and seeing the changes in your PDF. Make some different paragraphs and add variable spaces. Explore how your editor works; click on your source and find how to go to the same line in your PDF. Try adding some hard spaces and see how they influence line-breaking.

3

Mathematics Typing

This lesson presents LaTeX's math mode and how you can type inline and display formulas, the extensions provided by the `amsmath` package, and how to change fonts in math.

Typesetting complex mathematics is one of the greatest strengths of LaTeX. You can mark up mathematics in a logical way in what is known as 'math mode'.

3.1 Math mode

In math mode, spaces are ignored and the correct spacing between characters is (almost always) applied.

There are two forms of math mode:

- inline
- display

```
\documentclass{article}
\usepackage[T1]{fontenc}
\begin{document}
A sentence with inline mathematics: $y = mx + c$.
```

```
A second sentence with inline mathematics:
 ↪  $5^{2}=3^{2}+4^{2}$.


A second paragraph containing display math.
\[
  y = mx + c
\]
See how the paragraph continues after the display.
\end{document}
```

You may see 'LaTeX-like' mathematical input in other places, for example the MathJax system for placing equations in web pages. These systems often accept slight variations on LaTeX's syntax as they do not actually use LaTeX 'behind the scenes'.

Our examples are all *correct* LaTeX. If you see something different in another context, it might be because the example is not really using LaTeX.

3.1.1 Inline math mode and mathematical notation

As you can see above, inline math mode is marked using a pair of dollar symbols (`$...$`). It is also possible to use the notation `\( ... \)`. Simple expressions are entered without any special markup, and you'll see that the math is spaced out nicely and has letters in italic.

Inline math mode restricts vertical size of the expression so that as far as possible the formula does not disturb the linespacing of the paragraph.

Note that *all* mathematics should be marked up as math, even if it is a single character use `... $2$ ...` not `... 2 ...` otherwise, for example, when you need a negative number and need math to get a minus sign the `... $-2$ ...` may use math digits which may not be the same font as the text digits (depending on the document class). Conversely beware of math mode constructs appearing in plain text copied from elsewhere such as monetary values using $ or filenames using _ (which may be marked up as \$ and _ respectively).

We can easily add superscripts and subscripts; these are marked using ^ and _, respectively.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\begin{document}
Superscripts $a^{b}$ and subscripts $a_{b}$.
\end{document}
```

(You might see examples where simple super- and subscripts are entered without braces, but that is not the official syntax and can go wrong; always use braces.)

There are a *lot* of specialist math mode commands. Some of them are quite easy, for example `\sin` and `\log` for sine and logarithm or `\theta` for the Greek letter.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\begin{document}
Some mathematics: $y = 2 \sin \theta^{2}$.
\end{document}
```

We cannot cover all the standard LaTeX math mode commands here, but there are many online resources listing the standard set. You can look up commands for math mode symbols using the Detexify tool.

3.1.2 Display mathematics

You can use exactly the same commands for display math mode as for inline work. Display math mode is set centered by default and is meant for larger equations that are 'part of a paragraph'. Note that display math environments do not allow a paragraph to end within the mathematics, so you may not have blank lines within the source of the display.

The paragraph should always be started *before* the display so do not leave a blank line before the display math environment. If you need several lines of mathematics, do not use consecutive display math environments (this produces inconsistent spacing); use one of the multi-line display environments such as `align` from the `amsmath` package described later.

It's particularly useful for integrations, for example:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\begin{document}
A paragraph about a larger equation
\[
\int_{-\infty}^{+\infty} e^{-x^2} \, dx
\]
\end{document}
```

Notice here how sub-/superscript notation is used to set the limits on the integration.

We've added one piece of manual spacing here: `\,` makes a thin space before the `dx`. Formatting of the differential operator varies: some publishers use an upright 'd' whilst others use an italic '*d*'. One way to write your source to allow you to handle either is to create a command `\diff` that you can adjust as required, for example

```
\documentclass{article}
\usepackage[T1]{fontenc}
\newcommand{\diff}{\mathop{}\!d}            % For italic
% \newcommand{\diff}{\mathop{}\!\mathrm{d}} % For upright
\begin{document}
A paragraph about a larger equation
\[
\int_{-\infty}^{+\infty} e^{-x^2} \diff x
\]
\end{document}
```

You often want a numbered equation, which is created using the `equation` environment. Let's try the same example again:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\begin{document}
A paragraph about a larger equation
\begin{equation}
\int_{-\infty}^{+\infty} e^{-x^2} \, dx
\end{equation}
\end{document}
```

The equation number is incremented automatically and may be a simple number as in this example or may be prefixed by a section number, so (2.5) for the 5th equation in section 2. The details of the formatting are set up by the document class and not described here.

3.2 The amsmath package

Mathematical notation is very rich, and this means that the tools built into the LaTeX kernel can't cover everything. The amsmath package extends the core support to cover a lot more ideas. The amsmath User Guide contains many more examples than we can show in this lesson.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{amsmath}

\begin{document}
Solve the following recurrence for $ n,k\geq 0 $:
\begin{align*}
  Q_{n,0} &= 1   \quad Q_{0,k} = [k=0];  \\
  Q_{n,k} &= Q_{n-1,k}+Q_{n-1,k-1}+\binom{n}{k},
   ↪  \quad\text{for $n$, $k>0$.}
\end{align*}
\end{document}
```

The align* environment makes the equations line up on the ampersands, the & symbols, just like a table. Notice how we've used \quad to insert a bit of space, and \text to put some normal text inside math mode. We've also used another math mode command, \binom, for a binomial.

Notice that here we used align*, and the equation didn't come out numbered. Most math environments number the equations by default, and the starred variant (with a *) disables numbering.

The package also has several other convenient environments, for example for matrices.

```
\documentclass{article}
\usepackage[T1]{fontenc}
```

```
\usepackage{amsmath}
\begin{document}
AMS matrices.
\[
\begin{matrix}
a & b & c \\
d & e & f
\end{matrix}
\quad
\begin{pmatrix}
a & b & c \\
d & e & f
\end{pmatrix}
\quad
\begin{bmatrix}
a & b & c \\
d & e & f
\end{bmatrix}
\]
\end{document}
```

3.3 Fonts in math mode

Unlike normal text, font changes in math mode often convey very specific meaning. They are therefore often written explicitly. There are a set of commands you need here:

- `\mathrm`: roman (upright)
- `\mathit`: italic spaced as 'text'
- `\mathbf`: boldface
- `\mathsf`: sans serif
- `\mathtt`: monospaced (typewriter)
- `\mathbb`: double-struck (blackboard bold) (provided by the `amsfonts` package)

Each of these takes Latin letters as an argument, so for example we might write a matrix as

```
\documentclass{article}
\usepackage[T1]{fontenc}
```

```
\begin{document}
The matrix $\mathbf{M}$.
\end{document}
```

Note that the default math italic separates letters so that they may be used to denote a product of variables. Use `\mathit` to make a word italic.

The `\math..` font commands use fonts specified for math use. Sometimes you need to embed a word that is part of the outer sentence structure and needs the current text font, for that you can use `\text{...}` (which is provided by the `amsmath` package) or specific font styles such as `\textrm{..}`.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\begin{document}

$\text{bad use } size  \neq \mathit{size} \neq \mathrm{size} $

\textit{$\text{bad use } size \neq \mathit{size} \neq
 ↪  \mathrm{size} $}

\end{document}
```

3.4 Further amsmath alignments

In addition to the `align*` environment shown in the main lesson, `amsmath` has several other display math constructs, notably `gather` for multi-line displays that do not need alignment, and `multline` for splitting a larger single expression over multiple lines, aligning the first line to the left, and the last to the right. In all cases the `*` form omits the equation numbers by default.

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}

\usepackage{amsmath}

\begin{document}
```

```
Gather
\begin{gather}
  P(x)=ax^{5}+bx^{4}+cx^{3}+dx^{2}+ex +f\\
  x^2+x=10
\end{gather}

Multline
\begin{multline*}
   (a+b+c+d)x^{5}+(b+c+d+e)x^{4} \\
    +(c+d+e+f)x^{3}+(d+e+f+a)x^{2}+(e+f+a+b)x\\
    + (f+a+b+c)
\end{multline*}
\end{document}
```

3.4.1 Columns in math alignments

The amsmath alignment environments are designed to take pairs of columns with the first column of each pair aligned to the right and the second aligned to the left. This allows multiple equations to be shown, each aligned towards its relation symbol.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\begin{document}
Aligned equations
\begin{align*}
a &= b+1   &  c &= d+2  &  e &= f+3   \\
r &= s^{2} &  t &=u^{3} &  v &= w^{4}
\end{align*}

\end{document}
```

In addition there are variants of the display environments ending in ed that make a subterm inside a larger display. For example, aligned and gathered are variants of align and gather respectively.

```
\documentclass{article}
```

```
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\begin{document}
\begin{itemize}
\item
$\begin{aligned}[t]
a&=b\\
c&=d
\end{aligned}$
\item
$\begin{aligned}
a&=b\\
c&=d
\end{aligned}$
\end{itemize}
\end{document}
```

3.5 Bold Math

Standard LaTeX has two methods to give bold symbols in math. To make an entire expression bold, use `\boldmath` before entering the expression. The command `\mathbf` is also available to set individual letters or words in upright bold roman.

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}

\begin{document}


$(x+y)(x-y)=x^{2}-y^{2}$

{\boldmath $(x+y)(x-y)=x^{2}-y^{2}$ $\pi r^2$}

$(x+\mathbf{y})(x-\mathbf{y})=x^{2}-{\mathbf{y}}^{2}$
$\mathbf{\pi} r^2$ % bad use of \mathbf
\end{document}
```

If you want to access bold symbols (as would be used by `\boldmath`) within an otherwise normal weight expression, then you can use the command `\bm` from the

bm package. Note that \bm also works with symbols such as = and Greek letters. (Note that \mathbf has no effect on \pi in the example above.)

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{bm}

\begin{document}

$(x+\mathbf{y})(x-\mathbf{y})=x^{2}-{\mathbf{y}}^{2}$

$(x+\bm{y})(x-\bm{y}) \bm{=} x^{2}-{\bm{y}}^{2}$

$\alpha + \bm{\alpha} < \beta + \bm{\beta}$

\end{document}
```

3.6 Mathtools

The package mathtools loads amsmath and adds several additional features, such as variants of the amsmath matrix environments that allow the column alignment to be specified.

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{mathtools}

\begin{document}

\[
\begin{pmatrix*}[r]
  10&11\\
   1&2\\
  -5&-6
\end{pmatrix*}
\]

\end{document}
```

3.7 Unicode Math

There are variant TeX engines that use OpenType fonts. By default, these engines still use classic TeX math fonts but you may use the `unicode-math` package to use OpenType Math fonts. The details of this package are beyond this course and we refer you to the package documentation. However, we give a small example here.

```
% !TEX lualatex
\documentclass[a4paper]{article}
\usepackage{unicode-math}
\setmainfont{TeX Gyre Pagella}
\setmathfont{TeX Gyre Pagella Math}

\begin{document}

One two three
\[
\log \alpha + \log \beta = \log(\alpha\beta)
\]

Unicode Math Alphanumerics
\[A + \symfrak{A}+\symbf{A}+ \symcal{A} + \symscr{A}+
 ↪  \symbb{A}\]

\end{document}
```

3.8 Exercises

Try out some basic math mode work: take the examples and switch between inline and display math modes. Can you see what effect this has.

Try adding other Greek letters, both lower- and uppercase. You should be able to guess the names.

Experiment with the font changing commands: what happens when you try to nest them?

Displayed math is centered by default; try adding the document class option `[fleqn]` (flush left equation) option to some of the above examples to see a differ-

ent layout. Similarly equation numbers are usually on the right. Experiment with adding the `[leqno]` (left equation numbers) document class option.

4

Including Graphics

To bring in graphics from outside LaTeX, use the `graphicx` package, which adds the command `\includegraphics` to LaTeX.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{graphicx}

\begin{document}
This picture
\begin{center}
  \includegraphics[height=2cm]{example-image}
\end{center}
is an imported PDF.
\end{document}
```

You can include EPS, PNG, JPG, and PDF files. If you have more than one version of a graphic then you can write, for instance, `example-image.png`. (The `graphicx` package will try to guess the extension if you do not give one.)

You'll notice we've used a new environment here, `center`, to place the image horizontally centered on the page. A bit later, we'll talk more about spacing and positioning.

4.1 Altering graphic appearance

The `\includegraphics` command has many options to control the size and shape of the included images and to trim down material. Some of these are used a lot, so they are worth being aware of.

The most obvious thing to set is the `width` or the `height` of an image, which are often given relative to the `\textwidth` or `\linewidth` and `\textheight`. The difference between `\textwidth` and `\linewidth` is subtle and often the result is the same. `\textwidth` is the width of the text block on the physical page, whereas `\linewidth` is the *current* width, which might locally be different (the difference is most obvious with the class option `twocolumn`). LaTeX will automatically scale the image so that the aspect ratio stays correct.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{graphicx}

\begin{document}
\begin{center}
  \includegraphics[height = 0.5\textheight]{example-image}
\end{center}
Some text
\begin{center}
  \includegraphics[width = 0.5\textwidth]{example-image}
\end{center}
\end{document}
```

You can also `scale` images, or rotate them by an `angle`. The other thing you might want to do is to `clip` and `trim` an image.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{graphicx}

\begin{document}
\begin{center}
  \includegraphics[clip, trim = 0 0 50 50]{example-image}
\end{center}
\end{document}
```

4.2 Making images float

Traditionally in typesetting, particularly with technical documents, graphics may move to another spot in the document. This is called a *float*. Images are normally included as floats so they do not leave large gaps in the page.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{lipsum}  % produce dummy text as filler

\begin{document}
\lipsum[1-4] % Just a few filler paragraphs

Test location.
\begin{figure}[ht]
  \centering
  \includegraphics[width=0.5\textwidth]{example-image-a.png}
  \caption{An example image}
\end{figure}

\lipsum[6-10] % Just a few filler paragraphs
\end{document}
```

Here LaTeX moves the graphic and the caption away from the `Test location` text to the top of the second page, because there isn't room for it on the bottom of the first page. The `ht` influences where LaTeX can place the float; these two letters mean that it can go where it is in the source (next to `Test location`) or to the top of a page. You can use up to four position specifiers

- h 'Here' (if possible)
- t Top of the page
- b Bottom of the page
- p A dedicated page only for floats

You'll probably spot that we've centered the image here using `\centering` rather than the `center` environment. Inside a float, you should use `\centering` if you want to horizontally center content; this avoids both the float and `center` environment adding extra vertical space.

4.3 Naming graphics files

LaTeX works on many computer platforms so file names deserve some thought. Safest is to name your graphics simply, in particular without spaces. For example, if you want to organize your files by keeping all graphics in a subdirectory, then something like `\includegraphics[width=30pt]{pix/mom.png}` is portable and future-proof.

Spaces in file names are traditionally somewhat problematic, but are now generally supported. However, if you have spaces in the name, and you have issues, you may wish to try removing the spaces as the first step.

Accented character support is somewhat variable; there are issues with some systems, particularly on Windows. If you find issues with accented characters in file names, try using only ASCII characters for a test.

4.4 Storing graphics in a subdirectory

A common way to lay out source files is to put all graphics into a subdirectory. You can then include the relative path, as is shown above; notice that the / character is used to separate parts of the path even on Windows.

If you have a lot of graphics, you might want to set up the subdirectory in advance. That can be done using `\graphicspath`, which needs a braced entry for each subdirectory. For example, to include both `figs` and `pics` subdirectories, we would have:

```
\graphicspath{{figs/}{pics/}}
```

Notice in particular the trailing / in these.

4.5 Producing graphics

As discussed, LaTeX easily uses graphics from most sources, including plots from scientific software. When you do that, you probably want to save as a PDF if you can, as this is a scalable format. If you do need to create a bitmap, aim for high

resolution. You can make mouse-created graphics that include LaTeX snippets with Inkscape. An alternative that in addition extends those drawing techniques to three dimensions is Asymptote. These two produce their output as files that you include in your document.

You can also create graphics such as drawings that are especially suited to LaTeX, with very high precision as well as equations and labels that match your document. You can draw graphics directly inside your document, which is convenient although at the cost of more complex documents with larger requirements, by using Ti*k*Z. An alternative is PSTricks.

4.6 Placing floats

LaTeX's float placement is complex. The most common request is to have the figure placed in the output exactly where it lies in the input. The `float` package will do that.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{lipsum}  % dummy text for filler
\usepackage{float}

\begin{document}
\lipsum[1-7]
\begin{figure}[H]
  \centering
  \includegraphics[width=0.5\textwidth]{example-image}
  \caption{An example image}
\end{figure}
\lipsum[8-15]
\end{document}
```

Note the `H` option, which puts the figure 'absolutely Here'. However it is often not recommended to use `H`, because it may create large portions of white space in your document.

4.7 Other types of float

You might want to have other types of floating environment; each type is inserted independently. You can do that using the `trivfloat` package. This provides a single command, `\trivfloat`, to make new types of float.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage{lipsum}  % dummy text for filler
\usepackage{trivfloat}
\trivfloat{image}

\begin{document}
\begin{image}
  \centering
  \includegraphics[width=0.5\textwidth]{example-image}
  \caption{An example image}
\end{image}
\end{document}
```

4.8 Cross-referencing

When you are writing a document of any length, you'll want to refer to numbered items such as figures, tables or equations. Luckily, LaTeX can automatically add the right numbers; we just have to set things up.

4.8.1 The `\label` and `\ref` mechanism

To have LaTeX remember a spot in your document you have to label it, and then in other places, you refer to it.

```
\documentclass{article}
\usepackage[T1]{fontenc}

\begin{document}
```

```
Hey world!

This is a first document.

\section{Title of the first section}

Text of material for the first section.


\subsection{Subsection of the first section}
\label{subsec:labelone}

Text of material for the first subsection.
\begin{equation}
  e^{i\pi}+1 = 0
\label{eq:labeltwo}
\end{equation}

In subsection~\ref{subsec:labelone} is
 ↪  equation~\ref{eq:labeltwo}.
\end{document}
```

There are two `\label{...}` commands, one after the subsection and one inside the equation environment. They are associated with the last sentence's `\ref{...}` commands. When you run LaTeX, it saves information about the labels to an auxiliary file. For `\label{subsec:labelone}`, LaTeX knows that it is now in a subsection and so it saves the subsection's number. For `\label{eq:labeltwo}`, LaTeX knows that the most recent environment of interest is an equation so it saves the information for that equation. When you ask for the reference, LaTeX gets it from the auxiliary file.

The `subsec:` and `eq:` aren't used by LaTeX; rather, it just keeps track of what it has most recently processed. But when you are writing these help you remember what the label is about.

You may see references that show in an output PDF as boldface double question marks, **??**. The explanation is that because of this auxiliary file work, the first time that you compile a document the label has not yet been saved. Run LaTeX one more time and you'll be all set. (Usually while writing you will run LaTeX several times anyway, so in practice this is not a bother.)

Notice the tilde (~) characters before the references. You don't want a line break between `subsection` and its number, or between `equation` and its number. Putting

in a tilde means LaTeX won't break the line there.

4.8.2 Where to put `\label`

The `\label` command always refers to the previous numbered entity: a section, an equation, a float, etc. That means that `\label` always has to come *after* the thing you want to refer to. In particular, when you create floats, the `\label` has to come *after* (or better, in), the `\caption` command, but within the float environment.

4.8.3 Making cross-references into links

You can make your cross-references into hyperlinks using the `hyperref` package. In most cases, `hyperref` should be loaded after any other packages specified in the document preamble.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[hidelinks]{hyperref}
\begin{document}

\section{Introduction}
Some exciting text with a reference~\ref{sec:next}.

\section{Next thing}
\label{sec:next}

More text here.
\end{document}
```

We have chosen to make the links the same color as the normal text; try removing `hidelinks` to see why!

4.9 Exercises

- Try including an image you have created, replacing the 'standard' ones we have used in the demonstration.

- Explore what you can do using the `height`, `width`, `angle` and `scale` keys.
- Use the `width` key to set the size of a graphic relative to `\textwidth` and another graphic relative to `\linewidth`. Try out how they behave with or without the `twocolumn` option.
- Use `lipsum` to make a reasonably long demonstration, then try out placing floats using the different position specifiers. How do different specifiers interact?
- Try adding new numbered parts (sections, subsections, enumerated lists) to the test document and finding out how many runs are needed to make `\label` commands work.
- Add some floats and see what happens when you put `\label` before the `\caption` instead of after; can you predict the result?
- What happens if you put a `\label` for an equation after the `\end{equation}`?

5

Tables

5.1 The `array` package

Tables in LaTeX are set using the `tabular` environment.

We will assume you load the `array` package, which adds more functionality to LaTeX tables, and which is not built into the LaTeX kernel only for historic reasons. So put the following in your preamble and we're good to go:

```
\usepackage{array}
```

In order to typeset a `tabular` we have to tell LaTeX how many columns will be needed and how they should be aligned. This is done in a mandatory argument – often referred to as the table preamble – to the `tabular` environment, in which you specify the columns by using single-letter names, called preamble-tokens. The available column types are (see Table 5.1).

Table 5.1: The available column types

type	description
`l`	left aligned column
`c`	centered column
`r`	right aligned column

type	description
`p{width}`	a column with fixed width `width`; the text will be automatically line wrapped and fully justified
`m{width}`	like `p`, but vertically centered compared to the rest of the row
`b{width}`	like `p`, but bottom aligned
`w{align}{width}`	prints the contents with a fixed `width`, silently overprinting if things get larger. You can choose the horizontal alignment using `l`, `c`, or `r`.
`W{align}{width}`	like `w`, but this will issue an overfull box warning if things get too wide.

In addition, a few other preamble-tokens are available which don't define a column but might be useful as well (see Table 5.2).

Table 5.2: The other preamble-tokens

type	description
`*{num}{string}`	repeats `string` for `num` times in the preamble. With this you can define multiple identical columns.
`>{decl}`	this will put `decl` before the contents of every cell in the following column (this is useful, e.g., to set a different font for this column)
`<{decl}`	this will put `decl` after the contents of each cell in the previous column
`|`	add a vertical rule
`@{decl}`	replace the space between two columns with `decl`
`!{decl}`	add `decl` in the center of the existing space

These two tables list all the available column types from LaTeX and the `array` package.

The columns `l`, `c`, and `r` will have the natural width of the widest cell. Each column has to be declared, so if you want three centered columns, you'd use `ccc` in the table preamble. Spaces are ignored, so `c c c` is the same.

In a table body columns are separated using an ampersand `&` and a new row is started using `\\`.

We have everything we need for our first table. In the following code the `&` and `\\` are aligned. This isn't necessary in LaTeX, but helps reading the source.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{lll}
  Animal & Food  & Size   \\
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
\end{tabular}
\end{document}
```

If a table column contains a lot of text you will have issues to get that right with only `l`, `c`, and `r`. See what happens in the following example:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{cl}
  Animal & Description \\
  dog    & The dog is a member of the genus Canis, which forms
   ↪  part of the
          wolf-like canids, and is the most widely abundant
           ↪  terrestrial
          carnivore. \\
  cat    & The cat is a domestic species of small carnivorous
   ↪  mammal. It is the
          only domesticated species in the family Felidae and
           ↪  is often referred
          to as the domestic cat to distinguish it from the
           ↪  wild members of the
          family. \\
\end{tabular}
\end{document}
```

The issue is that the `l` type column typesets its contents in a single row at its natural width, even if there is a page border in the way. To overcome this you can use the `p` column. This typesets its contents as paragraphs with the width you specify as

an argument and vertically aligns them at the top – which you'll want most of the time. Compare the above outcome to the following:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{cp{9cm}}
  Animal & Description \\
  dog    & The dog is a member of the genus Canis, which forms
   ↪  part of the
          wolf-like canids, and is the most widely abundant
           ↪  terrestrial
          carnivore. \\
  cat    & The cat is a domestic species of small carnivorous
   ↪  mammal. It is the
          only domesticated species in the family Felidae and
           ↪  is often referred
          to as the domestic cat to distinguish it from the
           ↪  wild members of the
          family. \\
\end{tabular}
\end{document}
```

If your table has many columns of the same type it is cumbersome to put that many column definitions in the preamble. You can make things easier by using `*{num}{string}`, which repeats the `string` num times. So `*{6}{c}` is equivalent to `cccccc`. To show you that it works here is the first table of this lesson with the newly learned syntax:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{*{3}{l}}
  Animal & Food  & Size   \\
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
```

```
\end{tabular}
\end{document}
```

5.2 Adding rules (lines)

A word of advice prior to introducing rules; lines should be used really sparsely in tables, and normally vertical ones look unprofessional. In fact, for professional tables you shouldn't use any of the standard lines; instead you should get familiar with the facilities of the `booktabs` package, which is why it is covered here first.

`booktabs` provides four different types of lines. Each of those commands has to be used as the first thing in a row or following another rule. Three of the rule commands are: `\toprule`, `\midrule`, and `\bottomrule`. From their names the intended place of use should be clear:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}


\begin{document}
\begin{tabular}{lll}
  \toprule
  Animal & Food  & Size   \\
  \midrule
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
  \bottomrule
\end{tabular}
\end{document}
```

The fourth rule command provided by `booktabs` is `\cmidrule`. It can be used to draw a rule that doesn't span the entire width of the table but only a specified column range. A column range is entered as a number span: {*number*-*number*}. Even if you only want to draw the rule for a single column you need to specify that as a range (with both numbers matching).

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{lll}
  \toprule
  Animal & Food  & Size   \\
  \midrule
  dog    & meat  & medium \\
  \cmidrule{1-2}
  horse  & hay   & large  \\
  \cmidrule{1-1}
  \cmidrule{3-3}
  frog   & flies & small  \\
  \bottomrule
\end{tabular}
\end{document}
```

There is another useful feature of `\cmidrule`. You can shorten it on either end with an optional argument enclosed in parentheses:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{lll}
  \toprule
  Animal & Food  & Size   \\
  \midrule
  dog    & meat  & medium \\
  \cmidrule{1-2}
  horse  & hay   & large  \\
  \cmidrule(r){1-1}
  \cmidrule(rl){2-2}
  \cmidrule(l){3-3}
  frog   & flies & small  \\
  \bottomrule
\end{tabular}
```

```
\end{document}
```

You may have guessed that `r` and `l` mean the rule is shortened on its **r**ight and **l**eft end, respectively.

Sometimes a rule would be too much of a separation for two rows but to get across the meaning more clearly you want to separate them by some means. In this case you can use `\addlinespace` to insert a small skip.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{cp{9cm}}
  \toprule
  Animal & Description \\
  \midrule
  dog    & The dog is a member of the genus Canis, which forms
   ↪  part of the
          wolf-like canids, and is the most widely abundant
           ↪  terrestrial
          carnivore. \\
  \addlinespace
  cat    & The cat is a domestic species of small carnivorous
   ↪  mammal. It is the
          only domesticated species in the family Felidae and
           ↪  is often referred
          to as the domestic cat to distinguish it from the
           ↪  wild members of the
          family. \\
  \bottomrule
\end{tabular}
\end{document}
```

5.3 Merging cells

In LaTeX you can merge cells horizontally by using the `\multicolumn` command. It has to be used as the first thing in a cell. `\multicolumn` takes three arguments:

1. The number of cells which should be merged
2. The alignment of the merged cell
3. The contents of the merged cell

The alignment can contain anything legal in a `tabular`'s preamble, but *only a single column type*.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}


\begin{document}
\begin{tabular}{lll}
  \toprule
  Animal & Food  & Size   \\
  \midrule
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
  fuath  & \multicolumn{2}{c}{unknown} \\
  \bottomrule
\end{tabular}
\end{document}
```

You can also use `\multicolumn` on a single cell to prevent the application of whatever you defined in the table preamble for the current column. The following uses this method to center the table's head row:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}
```

```
\begin{document}
\begin{tabular}{lll}
  \toprule
  \multicolumn{1}{c}{Animal} & \multicolumn{1}{c}{Food} &
   ↪  \multicolumn{1}{c}{Size} \\
  \midrule
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
  fuath  & \multicolumn{2}{c}{unknown} \\
  \bottomrule
\end{tabular}
\end{document}
```

Merging cells vertically isn't supported by LaTeX. Usually it suffices to leave cells empty to give the reader the correct idea of what was meant without explicitly making cells span rows.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}


\begin{document}
\begin{tabular}{lll}
  \toprule
  Group     & Animal & Size   \\
  \midrule
  herbivore & horse  & large  \\
            & deer   & medium \\
            & rabbit & small  \\
  \addlinespace
  carnivore & dog    & medium \\
            & cat    & small  \\
            & lion   & large  \\
  \addlinespace
  omnivore  & crow   & small  \\
            & bear   & large  \\
            & pig    & medium \\
  \bottomrule
```

```
\end{tabular}
\end{document}
```

5.4 The other preamble contents

As the lesson didn't cover all the available preamble-tokens, a few others are explained with examples here. You might want to revisit the tables at the start of the lesson to get an overview of the things available. The short descriptions provided there should suffice to understand what the different column types `m`, `b`, `w`, and `W` do after you understood `l`, `c`, `r`, and `p`. If not you might want to experiment a bit with them. What's still missing are the handy other preamble-tokens `>`, `<`, `@`, `!`, and `|`.

5.4.1 Styling a column

Since `>` and `<` can be used to put things before and after the cell contents of a column, you can use these to add commands which affect the look of a column. For instance, if you want to italicize the first column and put a colon after it, you can do the following:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{>{\itshape}l<{:} *{2}{l}}
  \toprule
  Animal & Food  & Size   \\
  \midrule
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
  \bottomrule
\end{tabular}
\end{document}
```

`\itshape` makes all the following text italic, but its effect is 'contained' by the table cell.

You may want the first cell not to be affected because it is the table head. Here `\multicolumn` may be used. Remember that it can be used to change a single cell's alignment as shown below.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{>{\itshape}l<{:} *{2}{l}}
  \toprule
  \multicolumn{1}{l}{Animal} & Food  & Size   \\
  \midrule
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
  \bottomrule
\end{tabular}
\end{document}
```

5.4.2 Manipulating the space between columns

Usually LaTeX pads each column by some space on both sides to give a balanced look and separate them. This space is defined with the length `\tabcolsep`. Due to the fact that each column is padded on both sides you get one `\tabcolsep` on either end of the table, and `2\tabcolsep` between two columns – one from each column. You can adjust this space to any length using `\setlength`:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\setlength\tabcolsep{1cm}

\begin{document}
\begin{tabular}{lll}
```

```
  Animal & Food  & Size   \\
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
\end{tabular}
\end{document}
```

You can change this space to something arbitrary using `@`. This will remove the padding between two columns or on either end, and instead put anything in between the columns you specify as an argument:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{l@{ : }l@{\hspace{2cm}}l}
  Animal & Food  & Size   \\
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
\end{tabular}
\end{document}
```

The `!` preamble token does something pretty similar. The difference is, that it *adds* its argument in center of the space between two columns.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{l!{:}ll}
  Animal & Food  & Size   \\
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
\end{tabular}
\end{document}
```

5.4.3 Vertical rules

Sometimes you have to use vertical rules.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}

\begin{document}
\begin{tabular}{l|ll}
  Animal & Food  & Size   \\[2pt]
  dog    & meat  & medium \\
  horse  & hay   & large  \\
  frog   & flies & small  \\
\end{tabular}
\end{document}
```

You might notice that the behavior of `|` is pretty similar to `!{decl}`; it adds the vertical rule between two columns leaving the padding as it is. There is a huge downside to this though; vertical rules don't work with the horizontal rules provided by `booktabs`. You can use the horizontal rules provided by LaTeX; those are `\hline` (corresponding to `\toprule`, `\midrule`, and `\bottomrule`) and `\cline` (which behaves like `\cmidrule`). As shown above, vertical rules will span any space specified in the optional argument to `\\`.

5.5 Customizing booktabs rules

All the `booktabs` rules and also `\addlinespace` support an optional argument in brackets with which you can specify the rule's thickness. In addition the trimming provided by `\cmidrule` can be customized by specifying a length in braces after `r` or `l`.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
```

```
\begin{tabular}{@{} lll@{}} \toprule[2pt]
  Animal & Food  & Size   \\ \midrule[1pt]
  dog    & meat  & medium \\
  \cmidrule[0.5pt](r{1pt}l{1cm}){1-2}
  horse  & hay   & large  \\
  frog   & flies & small  \\ \bottomrule[2pt]
\end{tabular}
\end{document}
```

5.6 Numeric alignment in columns

The alignment of numbers in tables can be handled by the column type S that is provided by the `siunitx` package.

A simple example with two aligned numeric columns would be:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{booktabs}
\usepackage{siunitx}
\begin{document}
\begin{tabular}{SS}
\toprule
{Values} &  {More Values} \\
\midrule
1        &   2.3456 \\
1.2      &   34.2345 \\
-2.3     &   90.473 \\
40       &   5642.5 \\
5.3      &   1.2e3 \\
0.2      &    1e4 \\
\bottomrule
\end{tabular}
\end{document}
```

Note that any non-numeric cell must be “protected” by enclosing it in braces.

The `siunitx` package provides many possibilities for formatting the numbers in different ways; see the package documentation.

5.7 Specifying the total table width

The width of a `tabular` environment is automatically determined based on the contents of the table. There are two commonly used mechanisms to specify a different total width.

Note that it is almost always preferable to format the table to a specified width as below (perhaps using a font size such as `\small` if necessary) rather than scaling a table with `\resizebox` and similar commands which will produce inconsistent font sizes and rule widths.

5.7.1 tabular*

The `tabular*` environment takes an additional *width* argument that specifies the total width of the table. Stretchy space must be added to the table using the `\extracolsep` command. This space is added between all columns from that point in the preamble. It is almost always used with `\fill`, a special space that stretches to be as large as necessary.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\begin{document}

\begin{center}
\begin{tabular}{cc}
\hline
A & B\\
C & D\\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular*}{.5\textwidth}{@{\extracolsep{\fill}}cc@{}}
\hline
A & B\\
C & D\\
\hline
\end{tabular*}
```

```
\end{center}

\begin{center}
\begin{tabular*}{\textwidth}{@{\extracolsep{\fill}}cc@{}}
\hline
A & B\\
C & D\\
\hline
\end{tabular*}
\end{center}

\end{document}
```

5.7.2 tabularx

The `tabularx` environment, provided by the package of the same name, has a similar syntax to `tabular*` but instead of adjusting the inter-column space, adjusts the widths of columns specified by a new column type, `X`. This is equivalent to a specification of `p{...}` for an automatically determined width.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{tabularx}
\begin{document}

\begin{center}
\begin{tabular}{lp{2cm}}
\hline
A & B B B B B B B B B B B B B B B B B B B B B B B B B\\
C & D D D D D D D\\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabularx}{.5\textwidth}{lX}
\hline
A & B B B B B B B B B B B B B B B B B B B B B B B B B\\
C & D D D D D D D\\
\hline
\end{tabularx}
```

```
\end{center}

\begin{center}
\begin{tabularx}{\textwidth}{lX}
\hline
A & B B B B B B B B B B B B B B B B B B B B B B B B B\\
C & D D D D D D D\\
\hline
\end{tabularx}
\end{center}

\end{document}
```

Unlike the other forms discussed in these lessons, `tabularx` needs to typeset the table several times with trial widths to determine the final setting. This means that there are several restrictions on the use of the environment; see the package documentation.

5.8 Multi-page tables

A `tabular` forms an unbreakable box so it must be small enough to fit on one page, and is often placed in a floating `table` environment.

Several packages provide variants with similar syntax that do allow page breaking. Here we show the `longtable` package:

```
\documentclass{article}
\usepackage[paperheight=8cm,paperwidth=8cm]{geometry}
\usepackage{array}
\usepackage{longtable}
\begin{document}
\begin{longtable*}{cc}
\multicolumn{2}{c}{A Long Table}\\
Left Side & Right Side\\
\hline
\endhead
\hline
\endfoot
aa & bb\\
Entry & b\\
```

```
a & b\\
a & b\\
a & b\\
a & b\\
a & bbb\\
a & b\\
a & b\\
a & b\\
a & b\\
a & b\\
a & b\\
a & b b b b b b\\
a & b b b b b\\
a & b b\\
A Wider Entry & b\\
\end{longtable*}

\end{document}
```

`longtable` is notable in that it preserves the column widths over all pages of the table; however in order to achieve this it may take several runs of LaTeX so that wide entries encountered later in the table can affect the column widths in earlier pages.

5.9 Table notes

It is quite common to need footnote-like marks in a table referring to notes under the table. The `threeparttable` package simplifies the markup for such tables, arranging that the notes are set in a block the same width as the table. Refer to the package documentation for full details, but we show a simple example here.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{threeparttable}
\begin{document}

\begin{table}
\begin{threeparttable}
   \caption{An Example}
```

```
   \begin{tabular}{ll}
    An entry & 42\tnote{1}\\
    Another entry & 24\tnote{2}\\
   \end{tabular}
   \begin{tablenotes}
   \item [1] the first note.
   \item [2] the second note.
   \end{tablenotes}
\end{threeparttable}
\end{table}

\end{document}
```

5.10 Typesetting in narrow columns

The default line breaking settings assume relatively long lines to give some flexibility in choosing line breaks. The following example shows some possible approaches. The first table shows interword spacing stretched and TeX warns about Underfull lines. Using `\raggedright` usually avoids this problem but may leave some lines 'too ragged'. The `\RaggedRight` command from the `ragged2e` package is a compromise; it allows some raggedness in the line lengths, but will also hyphenate where necessary, as shown in the third table.

Note the use of `\arraybackslash` here, which resets the definition of `\\` so that it ends the table row.

An alternative technique, as shown in the fourth table, is to use a smaller font so that the columns are not so narrow relative to the text size.

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{ragged2e}
\begin{document}

\begin{table}

\begin{tabular}[t]{lp{3cm}}
One & A long text set in a narrow paragraph, with some more
 ↪  example text.\\
```

```
Two & A different long text set in a narrow paragraph, with
 ↪ some more  hard to hyphenate words.
\end{tabular}%
\begin{tabular}[t]{l>{\raggedright\arraybackslash}p{3cm}}
One & A long text set in a narrow paragraph, with some more
 ↪ example text.\\
Two & A different long text set in a narrow paragraph, with
 ↪ some more  hard to hyphenate words.
\end{tabular}%
\begin{tabular}[t]{l>{\RaggedRight}p{3cm}}
One & A long text set in a narrow paragraph, with some more
 ↪ example text.\\
Two & A different long text set in a narrow paragraph, with
 ↪ some more  hard to hyphenate words.
\end{tabular}

\footnotesize
\begin{tabular}[t]{lp{3cm}}
One & A long text set in a narrow paragraph, with some more
 ↪ example text.\\
Two & A different long text set in a narrow paragraph, with
 ↪ some more  hard to hyphenate words.
\end{tabular}

\end{table}

\end{document}
```

5.11 Defining new column types

The `array` package allows constructs such as `>{\bfseries}c` to denote a bold centered column. It is often convenient to define a new column type to encapsulate such use, for example

```
\newcolumntype{B}{>{\bfseries}c}
```

would allow the use of `B` in table preambles to specify a bold centered column.

5.12 Vertical tricks

Often, rather than making a cell span multiple rows it is better to instead have a single row in which some cells are split vertically by the use of nested `tabular` environments.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{lcc}
  \toprule
  Test & \begin{tabular}{@{}c@{}}A\\a\end{tabular} &
   ↪  \begin{tabular}{@{}c@{}}B\\b\end{tabular} \\
  \midrule
  Content & is & here \\
  Content & is & here \\
  Content & is & here \\
  \bottomrule
\end{tabular}
\end{document}
```

Note that you can control vertical alignment by an optional argument to the `tabular`; it supports the usage of `t`, `c`, or `b` for top, centered, or bottom aligned respectively and is used like this:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\usepackage{booktabs}

\begin{document}
\begin{tabular}{lcc}
  \toprule
  Test & \begin{tabular}[b]{@{}c@{}}A\\a\end{tabular} &
   ↪  \begin{tabular}[t]{@{}c@{}}B\\b\end{tabular} \\
  \midrule
  Content & is & here \\
```

```
  Content & is & here \\
  Content & is & here \\
  \bottomrule
\end{tabular}
\end{document}
```

5.13 Line spacing in tables

In the main lesson we demonstrated `\addlinespace` from the `booktabs` package, which is useful for adding extra space between specific lines.

There are two general parameters that control line spacing, `\arraystretch` and `\extrarowheight` (the latter from the `array` package).

```
\renewcommand\arraystretch{1.5}
```

will increase the baseline spacing by 50%.

Often, especially when using `\hline`, it is better just to increase the height of rows, without increasing their depth below the baseline. The following example demonstrates the `\extrarowheight` parameter.

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage{array}
\begin{document}


\begin{center}
\begin{tabular}{cc}
\hline
Square& $x^2$\\
\hline
Cube& $x^3$\\
\hline
\end{tabular}
\end{center}
```

```
\begin{center}
\setlength\extrarowheight{2pt}
\begin{tabular}{cc}
\hline
Square& $x^2$\\
\hline
Cube& $x^3$\\
\hline
\end{tabular}
\end{center}
\end{document}
```

5.14 Exercises

- Use the simple table example to start experimenting with tables.
- Try out different alignments using the `l`, `c` and `r` column types.
- What happens if you have too few items in a table row?
- How about too many?
- Experiment with the `\multicolumn` command to span across columns.

6

Working with bibliography

For bibliographic citations, while you can include reference sources directly in your document, usually you will get that information from one or more external files. Such a file is a database of references, containing the information in a processing-friendly format. Using one or more reference databases lets you re-use information and avoid manual formatting.

6.1 Reference databases

Reference databases are normally referred to as 'BibTeX files' and have the extension `.bib`. They contain one or more entries, one for each reference, and within each entry there are a series of fields. Let us look at an example.

```
@article{Thomas2008,
  author  = {Thomas, Christine M. and Liu, Tianbiao and Hall,
   ↪  Michael B.
             and Darensbourg, Marcetta Y.},
  title   = {Series of Mixed Valent {Fe(II)Fe(I)} Complexes
   ↪  That Model the
             {H(OX)} State of [{FeFe}]Hydrogenase: Redox
              ↪  Properties,
             Density-Functional Theory Investigation, and
              ↪  Reactivity with
```

```
            Extrinsic {CO}},
  journal = {Inorg. Chem.},
  year    = {2008},
  volume  = {47},
  number  = {15},
  pages   = {7009-7024},
  doi     = {10.1021/ic800654a},
}
@book{Graham1995,
  author    = {Ronald L. Graham and Donald E. Knuth and Oren
   ↪  Patashnik},
  title     = {Concrete Mathematics},
  publisher = {Addison-Wesley},
  year      = {1995},
}
```

This is an entry for an article and another for a book; these are by far the most common types. Each database entry type starts with `@`, as shown, and all of the information then sits within a brace pair.

The various fields we need are given in key-value format, apart from what is known as the 'key': the 'name' of the citation. You can use whatever you like, as it's just a label, but above we've chosen to use the name of an author plus the year: this is a common approach.

Exactly which fields you need to give depends on the type of entry, but most of these are quite obvious. You might notice that in the `author` field, each entry is separated by and. This is *essential*: the format of the *output* needs to know which author is which. You might also notice that in the article title, some entries are in an extra set of braces; these are there to prevent any case-changing being applied.

Editing `.bib` files by hand is rather tedious, so most people use a dedicated editor. JabRef is widely used and cross-platform, but there are several other interfaces available. If the reference contains a DOI (Digital Object Identifier), you may want to try doi2bib to easily get the BibTeX entry. But make sure to check if the entry is correct!

Here, we will use the short example database above for our demonstrations: we have 'saved' it as `learnlatex.bib`.

6.2 Transferring information from the database

To get the information into your document there are three steps. First, use LaTeX to compile your document, which creates a file with a list of the references that your document cites. Second, run a program that takes information from the database of references, picks out the ones that you use, and puts them in order. Finally, compile your document again so that LaTeX can use that information to resolve your citations. Usually it will require at least two compilations to resolve all the references.

For the second step, there are two systems in wide use: BibTeX and Biber. Biber is only ever used with a LaTeX package called `biblatex`, whereas BibTeX is used with either no packages at all or with `natbib`.

Running a second tool as well as LaTeX is handled in different ways by different editors. For our online examples, there are some 'behind the scenes' scripts that do everything in one go. Your editor might have a single 'do stuff' button or you might have to choose to run BibTeX or Biber manually between LaTeX runs.

The format of citations and references is independent of your BibTeX database, and is set by what is known as a 'style'. We will see that these work slightly differently in the BibTeX workflow and `biblatex`, but the general idea remains: we can choose how citations appear.

6.3 The BibTeX workflow with `natbib`

Whilst it is possible to insert citations into a LaTeX document without any packages loaded, this is rather limited. Instead, we will use the `natbib` package, which allows us to create different types of citation and has a lot of styles available.

The basic structure of our input is as shown in this example.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{natbib}

\begin{document}
The mathematics showcase is from \citet{Graham1995}, whereas
there is some chemistry in \citet{Thomas2008}.
```

```
Some parenthetical citations: \citep{Graham1995}
and then \citep[p.~56]{Thomas2008}.

\citep[See][pp.~45--48]{Graham1995}

Together \citep{Graham1995,Thomas2008}

\bibliographystyle{plainnat}
\bibliography{learnlatex}
\end{document}
```

You can see that we can cite different entries in the database by giving their key. The `natbib` package offers both textual and parenthetical citation styles, `\citet` and `\citep`, respectively. The reference style is selected by the `\bibliographystyle` line; here we've used the `plainnat` style. The bibliography is actually inserted by the `\bibliography` line, which also picks the database(s) to use; this is a comma-separated list of names.

Page references can be added to the citation with an optional argument. If two optional arguments are given, the first goes in front of the citation label for a short note and the second after the label for a page reference.

The setup above uses author-year style, but we can make use of numeric citations. That is done by adding the `numbers` option to the `natbib` line.

6.4 The biblatex workflow

The `biblatex` package works slightly differently to `natbib`, as we select the databases in the preamble but print it in the document body. There are some new commands for this.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[style=authoryear]{biblatex}
\addbibresource{learnlatex.bib} % file of reference info

\begin{document}
The mathematics showcase is from \autocite{Graham1995}.
```

```
Some more complex citations: \parencite{Graham1995} or
\textcite{Thomas2008} or possibly \citetitle{Graham1995}.

\autocite[56]{Thomas2008}

\autocite[See][45-48]{Graham1995}

Together \autocite{Thomas2008,Graham1995}

\printbibliography
\end{document}
```

Notice that `\addbibresource` *requires* the full database filename, whereas we omitted the `.bib` for `\bibliography` with `natbib`. Also notice that `biblatex` uses rather longer names for its citation commands, but these are all quite easy to guess.

Again, short text before and after the citation can be inserted with the optional arguments. Note that the page numbers need not be prefixed with `p.~` or `pp.~` here, `biblatex` can automatically add the appropriate prefix.

In `biblatex`, the reference style is picked when we load the package. Here, we've used `authoryear`, but there is a `numeric` style and many others are also available.

6.5 Choosing between the BibTeX and BibLaTeX

Even though both the BibTeX workflow and `biblatex` get their input via BibTeX files and can produce structurally similar output in the document, they use completely different ways to produce this result. That means that there are some differences between the two approaches that may help you choose which one works best for you.

In the BibTeX workflow the bibliography style is ultimately decided by a `.bst` file which you select with the `\bibliographystyle` command. `biblatex` does not use `.bst` files and uses a different system. If you are using a template that comes with a `.bst` file or are given a `.bst` file for your project, you must use the BibTeX workflow and cannot use `biblatex`.

The different approach `biblatex` takes implies that you can modify the output of the bibliography and citation commands directly from your document preamble

using LaTeX-based commands. Modifications of BibTeX `.bst` styles on the other hand usually require working with these external files and need knowledge of the BibTeX programming language. Generally speaking, `biblatex` is said to be easier to customize than the BibTeX workflow.

In `biblatex` it is generally easier to implement more elaborate citation styles with a wider array of different citation commands. It also offers more context-dependent features. Roughly speaking this is less interesting for the styles common in many STEM subjects, but becomes relevant for some more complex styles in some areas of the humanities.

BibTeX can only sort US-ASCII characters correctly and relies on workarounds to provide US-ASCII-based sorting for non-US-ASCII characters. With Biber `biblatex` offers full Unicode sorting capabilities. Thus `biblatex` is usually a better choice if you want to sort your bibliography in a non-ASCII/non-English order.

Having been around for much longer than `biblatex`, the BibTeX workflow is more established than `biblatex`, meaning that many publishers and journals expect bibliographies generated via the BibTeX workflow. Those publishers cannot or generally do not accept submissions using `biblatex`.

The bottom line is: Check the author/submission guidelines if you are submitting to a journal or publisher. If you are given a `.bst` file, you must use the BibTeX workflow. If you want a relatively simple bibliography and citation style and only need English US-ASCII-based sorting, the BibTeX workflow should suffice. If you need a more complex citation style, non-English sorting or want easier access to citation and bibliography style customisation features, you will want to look into using `biblatex`.

6.6 Dealing with non-English sorting

The BibTeX program was written primarily to deal with references in English. It is very limited in handling accented characters, and even more limited with non-Latin letters. In contrast, the Biber program was written from the start to handle a mix of scripts properly.

This means that if you are sorting your bibliography, and you need to sort in anything other than English order, you really should be using `biblatex` and Biber, rather than `natbib` and BibTeX.

6.7 Hyperlinks

If you load the `hyperref` package, it will automatically make some content in your bibliography into links. This is particularly useful for URLs and DOIs.

6.8 Differences in best practice for BibTeX input l

While the overall syntax of the BibTeX files is the same whether you use the BibTeX workflow or `biblatex`, the set of fields that is supported (used by the style) and their exact meaning may not only vary between the BibTeX workflow and `biblatex`, but also between different BibTeX styles. A large 'core set' of entry types and fields is the same for almost all styles, but there are differences in some fields.

A common example is the URL. Some older BibTeX `.bst` styles (most notably the 'standard BibTeX styles', e.g. `plain.bst`, `unsrt.bst`, ...) predate the invention of the URL and have no dedicated field for the URL of an online resource. Many newer styles *do* have a dedicated `url` field. The workaround to show the URL in the older styles is usually to use the `howpublished` field, but with the newer styles it is of course preferable to use the dedicated `url` field.

In order to be able to make use of the full potential of your used style you will have to find out the set of fields it supports and their semantic.

6.9 Exercises

- Try out both the `natbib` and `biblatex` examples. For `natbib`, you'll need to run LaTeX, BibTeX, LaTeX, LaTeX; for `biblatex`, it's LaTeX, Biber, LaTeX.
- See what happens when you create new database entries and new citations.
- Add a citation that's not in the database and see how it appears.
- Experiment with `natbib`'s `numeric` and `biblatex`'s `style=numeric` option.

7

LaTeX presentations

7.1 Presentation with Beamer

In LaTeX it is possible to make presentations using the document class `beamer`.

7.1.1 Structure of a presentation

An empty presentation looks as follows:

```
\documentclass{beamer}

\usetheme{Copenhagen}
\author{Bert}
\title{A tale of two primes}

\begin{document}

\end{document}
```

To make slides you can use the `frame` environment with the title of your slide as the single argument. For example one could write (inside the `document` environment):

```
\begin{frame}
\titlepage
\end{frame}

\begin{frame}{Article}
Some text about the article.
\end{frame}

\begin{frame}{Mathematica}
A helpful tool for mathematicians.
\end{frame}
```

This gives a title slide and two slides with a header and some text.

To order the information in your presentation you can use the `block`, `columns`, `enumerate` and `itemize` environments. The `block` environment is a `beamer` specific environment which can be used in the following way:

```
\begin{frame}{Article}

\begin{block}{Example}
This is an example of a block.
\end{block}

\begin{block}{Euclid's theorem}
This is a theorem.
\end{block}

\end{frame}
```

The block's header can be removed by adding an empty group `{}` after `\begin{block}`.

7.1.2 Pauses

If you want the elements of your slide to appear one by one you can use `\pause`. This command can be placed almost anywhere in the code. For example between two `block` environments:

```
\begin{frame}{Article}

\begin{block}{Definition}
This is a definition.
\end{block}

\pause

\begin{block}{Euclid's theorem}
This is a theorem.
\end{block}

\end{frame}
```

or in an enumerate

```
\begin{frame}{Title}

\begin{enumerate}
\item Element
\pause
\item Element
\pause
\item Element
\end{enumerate}

\end{frame}
```

7.1.3 Uncover

Using the \uncover command you can precisely determine when each part of the slide will appear. This command gives more flexibility than the \pause command. Below an example how the command \uncover can be used

```
\begin{frame}
\frametitle{Sets}
A \alert{set} is a collection of objects.\uncover<2->{ For
 ↪  example:
```

```
\[
Z=\{\text{cow},\text{pig},\text{elephant}\}.
\]
\uncover<3->{We call the objects in $Z$ the \alert{elements}
 ↪ of $Z$.}\uncover<4->{ We write
\[
\text{cow} \in Z
\]}
\uncover<5->{with \`\`cow is an element of
 ↪ $Z$''.}\uncover<6->{ Frequently encountered sets are}
\[
\begin{split}
\uncover<7->{\mathbb{N}} \uncover<8->{= \{1,2,3,\ldots
 ↪ \}}&\uncover<9->{ \qquad (\text{\`\`natural numbers''})\\}
\uncover<10->{\mathbb{Z}} \uncover<11->{=
 ↪ \{\ldots,-2,-1,0,1,2,\ldots \}}&\uncover<12->{ \qquad
 ↪ (\text{\`\`integer numbers''})\\}
\uncover<13->{\mathbb{Q}} \uncover<14->{= \{p/q :
 ↪ p,q\in\mathbb{Z} \text{ and } q\neq 0\}
 ↪ }&\uncover<15->{\qquad (\text{\`\`rational numbers''})\\}
\uncover<16->{\mathbb{R}} \uncover<17->{= \{\hbox{decimal
 ↪ numbers}\}}&\uncover<18->{\quad\qquad (\text{\`\`real
 ↪ numbers''})}
\end{split}
\]
\end{frame}
```

It is also possible to use `\uncover` in the `align` environment. For example

```
\begin{frame}
    The derivative of $f(x) = g(x) \cdot h(x)$, with $g(x) =
     ↪ x^2$ and $h(x) = \sin(x)$ equals
    \begin{align*}
        f'(x) \uncover<2->{&= g'(x) \cdot h(x) +}
         ↪ \uncover<3->{g(x) \cdot h'(x)} \\
        &\uncover<4->{= 2x \cdot \sin(x) +}
         ↪ \uncover<5->{x^2\cdot \cos(x).}
    \end{align*}
\end{frame}
```

The enumeration in the angle brackets in the `\uncover` command indicates the order at which these will appear on the slide, i.e. `\uncover<1->` will appear first

and `\uncover<10->` will appear as tenth. This enumeration holds within a specific frame, in the next frame it resets and you can start again from `\uncover<1->`. Note that `\uncover<1-3>` means the specific content will appear only on the first three slides, after that it will disappear again.

Within an `\itemize` environment you can also indicate the order at which the various items will appear, this is done as follows:

```
\begin{itemize}
      \item<4-> $[a,b]\uncover<5->{=\{x\in\mathbb{R} : a\leq
       ↪ x\leq b\}$,}
      \item<6-> $(a,b)\uncover<7->{=\{x\in\mathbb{R} : a< x<
       ↪ b\}$,}
      \item<8-> $(a,\infty)\uncover<9->{=\{x\in\mathbb{R} :
       ↪ x>a\}$.}
 \end{itemize}
```

7.1.4 Layout

Previously we used the beamer theme `Copenhagen`. More themes can be found in the Beamer Theme Matrix. Try adding the following lines in the preamble.

```
\usetheme{Warsaw}
\usecolortheme{beaver}
```

7.1.5 Tips for a short presentation

7.1.5.1 Isolate the main subject and make sure to cover it early in your presentation.

Try to deliver one message to your fellow students; if you succeed doing just that, your presentation is already a success. If you want your audience to remember something, say it early on in your presentation.

7.1.5.2 Use pictures instead of text

A presentation is a story that the presenter tells the audience. The `beamer` slides only serve to support that story. Don't write down everything that you want to say, filter the content. Try to replace text by pictures where possible.

7.1.5.3 Examples instead of abstract definitions

Abstract definitions can be hard for the audience to grasp immediately. Illustrate definitions using examples. Sometimes it is convenient to give an example before stating the formal definition.

7.1.5.4 Watch the time

You only get a short time. Keep this in mind when creating the presentation and make sure to practise it to ensure its length is good. Also keep the length in mind when choosing the number of frames.

7.2 Posters

There are many different ways to get the general structure of a poster into LaTeX. In this lesson we will discuss the three most prevalent methods, which are:

1. the `a0poster` documentclass;
2. the `beamerposter` package for the `beamer` documentclass;
3. the `tikzposter` documentclass.

We will briefly discuss the main differences in user experience and output between these three methods, as well as give a short tutorial on how to get started with each method.

7.2.1 Main differences between the three methods

The `a0poster` documentclass is the most similar to the writing of a document in the `article` documentclass with which you are already familiar. The main

differences are as follows; firstly you will need to make use of a package which allows the text to be divided into columns, which flow naturally into each other, secondly the slightly different structure needed to present figures and tables. You can use section titles as you would in any LaTeX article, and you will have slightly more space for content. The main drawback of `a0poster` is the relative simplicity of the poster layout (especially for beginners at poster making). You can change the colours of the text, and play with the font and fontsizes, but it is much harder to incorporate things like special section headers, or a nice title box.

The `beamerposter` package is a package that can be used within the `beamer` documentclass. This means that you can use the same design elements as you would in a beamer presentation. Most crucially this means that you can use the beamerthemes that are available, use the UvA beamertheme, or even design a theme for yourself (though this is quite complicated for LaTeX beginners). You can also use the block, exampleblock, and alertblock environments you may be familiar with. You can use the columns environment to split your poster up into columns. Differently from the columns in `a0poster` these do not flow into each other. This has the benefit that you can easily decide what goes in which column, though it is more cumbersome to work with. The mayor drawback of `beamerposter` is the fact that the poster is started in the centre of the page and builds out from there. So if you do not have an entirely filled poster, this may look a bit strange.

The `tikzposter` documentclass probably takes the most getting used to. It is fairly similar to `beamerposter` in look, though it generates a nice title block via the command `maketitle`, which looks good but does make it difficult to include an image in the header. The structure of the poster is again worked in columns, like the ones in `beamerposter` though the syntax for typesetting them is slightly different. All of the content must be placed in blocks, for which the built in theme options have a nice layout. You can also add notes on top of your blocks. Just like in `a0poster` the method for including figures and tables is a little different than you are used to, though not at all complicated.

7.2.2 The `a0poster` documentclass

For a poster created in the documentclass `a0poster` we start our document with the following code:

```
\documentclass[a0, portrait]{a0poster}
```

The options given to the documentclass here define the size of your poster and its

orientation (landscape or portrait).

The next step is to load any packages that you would normally need, think of `babel`, `graphicx`, `tikz`, etc.

Next up you need to load the package `multicol`, which you will need to split your poster into columns. This should be done like this:

```
\usepackage{multicol}
\columnsep=100pt
```

If you would like the columns to be closer together, or further apart, you can change the value of `columnsep`.

You can now start the document as usual with `\begin{document}...\end{document}`.

To create a header for your poster with a title and an institute logo (or another image), you can use the `minipage` environment. Here is an example of code for such a header.

```
\begin{minipage}{.7\textwidth}
  \VeryHuge Look I'm making a poster \\ [0.75cm]
  \Large Ostap S. Bender \\
  \Large RUDN University
\end{minipage}
%
\begin{minipage}{.3\textwidth}
  \includegraphics[]{instituteLogo.png}
\end{minipage}
```

You can use commands like `\bf` or `\textbf{...}` to print (parts of) the header in bold type. You can change the size of the font by playing with the commands like `Large` and `VeryHuge`.

To divide your content into columns in `a0poster` we need to use the package `multicol` as shown above. Inside the document at the moment where we would like our columns to start, we use the following command:

```
\begin{multicols}{2}
```

```
  ...
\end{multicols}
```

The number in the braces at the top gives the number of columns the text will be divided into. The entire content of your poster can now be typeset inside this environment. The text will be automatically split up into two columns.

In `a0poster` you can typeset an abstract, bibliography, and sections (with or without numbers) as you usually would in a `.tex` file with the documentclass `article`.

Figures such as images, tables, or tikzpictures can be inserted into the poster. However, you cannot make use of the `figure` environment. Instead, we need to place any figures into a `center` environment like so:

```
\begin{center}
  \includegraphics[options]{filename.extension}
  \captionof{figure}{Insert your caption here}
\end{center}
%
\begin{center}
  \begin{tikzpicture}
    ...
  \end{tikzpicture}
  \captionof{figure}{...}
\end{center}
```

Note that we use the command `\captionof{}{}` here. This command can only work if the package `caption` has been loaded in the preamble. The first argument for `\captionof` is the type of content you are writing a caption for, in this case a figure, but this could also be a table for instance.

If you would like to play with the colour of your text you can do so by including `\usepackage[svgnames]{xcolor}` into your preamble. This will give you access to many colours that have already been named. You can then use the following commands to make parts of your text another colour.

```
Here follows some regular text, \color{BlueViolet} from now on
 ↪  the text has changed colour, \color{Black} and then we are
 ↪  back to normal.
```

The very handy Reference Guide gives the full list of colours and their svgnames.

7.2.3 The beamerposter package for the beamer documentc

A poster created with the beamerposter package needs to be created in the beamer documentclass. We start our file with the following command:

```
\documentclass[xcolor={svgnames}]{beamer}
```

Note that the option given to the beamer documentclass here is not strictly necessary but does enrich your poster as it gives access to about 150 colours to use.

Next step is to load any packages that you would normally need, thick of babel, graphicx, tikz, etc. Next up you get to choose your beamertheme and colour. You do this by using these commands:

```
\usetheme{...}
\usecolortheme{...}
```

You can choose which theme and colour you would like from the very handy Beamer Theme Matrix. You can then fill in the names of the theme an colour you choose on the dots.

The next and **most important** step, is to load the package that will turn your beamer file into a poster.

```
\usepackage[orientation=portrait,size=a0,scale=1.4]{beamerpost
↪ er}
```

These options passed to the package determine whether the poster is in portrait or landscape mode, how big the poster is, and how much the fonts are sized up.

After this step you only need to define the title etc, so that this can be inserted correclty into the margins of your poster. For example:

```
\title{Look I'm making a poster}
```

```
\author{Ostap S. Bender}
\institute{RUDN University}
```

Now you are ready to start the actual document. You do this, like always, by beginning the document. But in the case of a poster in beamer you also need to start the one and only frame of your beamer document.

```
\begin{document}
  \begin{frame}
    ...
  \end{frame}
\end{document}
```

The way in which you create columns in `beamerposter` is to use the columns environment. You can make several different amounts and sizes of columns, each under (or included) in the others. Every time the syntax is similar. Below follows an example for three columns of equal width, below whcih we have two columns of varying widths.

```
\begin{columns}
  \begin{column}{.33\textwidth}
    ...
  \end{column}
  %
  \begin{column}{.33\textwidth}
    ...
  \end{column}
  %
  \begin{column}{.33\textwidth}
    ...
  \begin{column}
\end{columns}
%
\begin{columns}
  \begin{column}{.7\textwidth}
    ...
  \end{column}
  %
  \begin{column}{.3\textwidth}
    ...
  \end{column}
```

```
\end{columns}
```

With these columns (or without if you would like your text to stretch across the entire poster) you can insert pieces of text, figures, tikzpictures, tables, blocks, bibliography, etc. You can do this all in the usual beamer way.

If you would like to play with the colours of your text you can use the following commands to make parts of your text another colour.

```
Here follows some regular text, \color{BlueViolet} from now on
↪ the text has changed colour, \color{Black} and then we are
↪ back to normal.
```

Note that you do need to have the option `xcolor={svgnames}` in your documentclass to be able to use these colours and many others. Please see the Reference Guide for the full list of colours and their svgnames.

7.2.4 The `tikzposter` documentclass

For a poster created with the `tikzposter` documentclass we start our file with the following command:

```
\documentclass[24pt, a0paper, portrait]{tikzposter}
```

The options given to the documentclass here, define the fontsize of your text, the size of the paper, and the orientation of the poster (landscape or portrait).

Next step is to load any packages that you would normally need, think of `babel`, `graphicx`, `tikz`, etc.

Next up you get to choose your preferred theme. You do this by using this command:

```
\usetheme{...}
```

You can choose which theme and colour you would like from the very handy Tikzposter Reference Guide. You can then fill in the name of the theme you choose on the dots.

You can now define the title, etc. In `tikzposter` the `\maketitle` command is how we make the header, so we need this information to be able to do that.

```
\title{Look I'm making a poster}
\author{Ostap S. Bender}
\institute{RUDN University}
```

Now you are ready to start the actual document. You do this, like always, by beginning the document. You can also immediately make the title header.

```
\begin{document}
  \maketitle

  ...
\end{document}
```

The way in which you create columns in `tikzposter` is to use the columns environment, just like for `beamerposter`. However, each column itself is not defined by using `\begin{column}{width}`. Instead you begin a column with a `\column{width}` command. In `tikzposter` you cannot nest these types of columns together. If you want to split a column into columns for a while, you will need to use the `multicol` package. Below you can see an example for three columns of equal width, below which we have two columns of varying widths.

```
\begin{columns}
  \column{.33}
    ...
  %
  \column{.33}
    ...
  %
  \column{.33}
    ...
\end{columns}
%
\begin{columns}
```

```
  \column{.7}
    ...
  %
  \column{.3}
    ...
\end{columns}
```

Within these columns (or without if you would like your text to strech across the entire poster) you can insert pieces of text, figures, tikzpictures, tables, bibliography, etc.

In `tikzposter` any content that you want to include on the poster much be inserted in a block. This is how to do that:

```
\block{title}{content}
```

Your block does not need to have a title. If you'd rather it didn't you can leave the first set of braces empty. A fun feature of `tikzposter` is the ability to add notes on a block. This can be done as follows:

```
\block{title}{content \vspace{height}}
\note[targetoffsetx=size, targetoffsety=size, width=size]{note
 ↪  content}
```

Here the command `\vspace{height}` creates a given amount of vertical whitespace so that the note will nicely fit in the block environment and not cover up the text. The first two options for the `\note` command are used to place the note exactly where you would like it in the block (i.e. in the bit of whitespace we generated with the `vspace` command), and the `width` option is used to define the width of the note.

Figures such as images, tables, or tikzpictures can be inserted into the poster. However, you cannot make use of the `figure` environment. Instead, we need to place any figures into a `center` environment like so:

```
\begin{center}
  \includegraphics[options]{filename.extension}
  \captionof{figure}{Insert your caption here}
\end{center}
```

```
%
\begin{center}
  \begin{tikzpicture}
    ...
  \end{tikzpicture}
  \captionof{figure}{...}
\end{center}
```

Note that we use the command `\captionof{}{}` here. This command can only work if the package `caption` has been loaded in the preamble. The first argument for `captionof` is the type of content you are writing a caption for, in this case a figure, but this could also be table for instance.

If you would like to play with the colours of your text you can use the package `color` to define your own colours by including a command in the preamble, i.e. `\definecolor{MyPink}{RGB}{194, 19, 182}` and then using the following commands to make parts of your text another colour.

```
Here follows some regular text, \color{MyPink} from now on the
 ↪  text has changed colour, \color{black} and then we are
 ↪  back to normal.
```

Note that `tikzposter` does not support the use of the `svgnames` option. That is why we need to predefine our own colours (`tikzposter` does come with several colours and their names included, though this is not an extensive range).

As you may have spotted, the colours we predefined ourselves are given in RGB form. It is very easy to find the RGB form of any colour, simply by Googling the phrase *colour picker*, and using Google's colour tool.

8

Diagrams and drawings as code

8.1 TikZ

There is a variety of different toolsets for creating graphical objects in a TeX document. The most well known and probably most widely used of these toolsets is Ti*k*Z, which is a recursive acronym for "Ti*k*Z ist *kein* Zeichenprogramm" (German for Ti*k*Z is not a drawing programme). From the name of the toolset you can see that generating a figure in Ti*k*Z will not work through drawing the figure like you might be familiar with doing in Paint or similar drawing programmes. To create a figure using Ti*k*Z you will need to code it by using specific TeX commands. We will take a look at the basics of Ti*k*Z here.

Three useful references and sources of inspiration for creating Ti*k*Z figure are:

- The Wikibooks entry on Ti*k*Z: https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ.
- The CTAN entry on Ti*k*Z with the official handbook and other documentation: https://www.ctan.org/pkg/pgf.
- The TeXample Ti*k*Z database full of examples and inspiration: https://texample.net/tikz/.

8.1.1 Building blocks of Ti*k*Z

To create a figure in Ti*k*Z you will first need to load the correct package in your TeX file, i.e. use `\usepackage{tikz}` in the preamble. There is a wide range of options and special Ti*k*Z libraries which can also be used and would need to be loaded specifically, but we will not need these in this beginner course. You can look at the available options and try them out yourself using the three references given above.

A Ti*k*Z figure is usually placed in a `\begin{figure}` LaTeX environment, as you would do with other figures. This is useful because it will allow you to give the figure a number and caption, so that you will be able to reference it easily. Within your `figure` environment you can start a Ti*k*Z figure by writing the code `\begin{tikzpicture}`, after which you can start typesetting your figure. As usual you close the environment with the code `\end{tikzpicture}`.

Unlike you are used to when typesetting text and mathematics in LaTeX, you will need one character which does not correspond directly to something you can see in your document. **Every line of Ti*k*Z code must be closed with a semicolon.**

Let us get started by typesetting a graph, step by step, in Ti*k*Z. At the end of this lesson we will be able to typeset the following figure.

8.1.2 Drawing lines

A graph consists of several points and lines. We start the drawing of a graph by defining the coordinates of the points. In Ti*k*Z this can be done in two different ways. A cartesian coordinate is given by two numbers (an x- and a y-coordinate) separated by a comma, and surrounded by round brackets, so `(-2,0)` or `(1.5,1.5)`. In Ti*k*Z coordinates can also be given in polar form. This also works by giving two numbers inside round brackets (the angle in degrees, and the length), this time separated by a colon, so `(180:2)` or `(45:2.1)` for (approximately) the same two coordinates. **Note:** when no unit of measurement is given the default is centimeters, coordinates can also be given as `(1cm, 2pt)` for example.

We can now draw direct straight lines between these coordinates using the `\draw` command.

```
\begin{tikzpicture}
  \draw (-1,0) -- (3,10pt) -- (35:3);
```

```
\end{tikzpicture}
```

The result of this code is the following image.

Apart from the drawing of direct straight lines, we can also define other types of lines, and these lines can be given colour, line thickness, or pattern. For example, use `-|` or `|-` in stead of `--` to draw lines that go horizontal first and then vertical, or the other way around. The style of the lines can be changed by giving options to the `\draw` command. This is done by putting these options in square brackets behind the command. In the following example we draw an arrow in stead of a line, and give another line a colour.

```
\begin{tikzpicture}
  \draw[->] (-1,0) -| (3,10pt);
  \draw[red] (3,10pt) -- (35:3);
\end{tikzpicture}
```

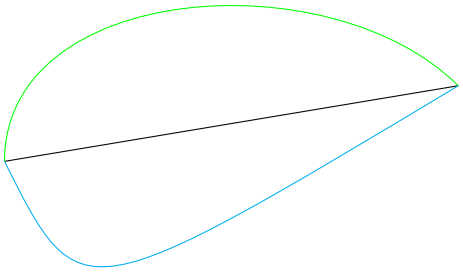
The result of this code is the following image.

Use the Ti*k*Z sources to see which types of options are possible.

We can not only draw straight lines, but also curved lines. This can be done in several different ways. In the following example we present three options.

```
\begin{tikzpicture}
  \draw (-1,0) to (5,1);
  \draw[green] (-1,0) to[out=90,in=135] (5,1);
  \draw[cyan] (-1,0) .. controls (0,-2) .. (5,1);
\end{tikzpicture}
```

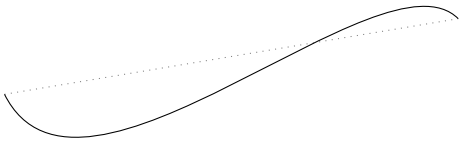
The result of this code is the following image.



The first thing that you might notice is the use of `to` in stead of `--` to connect the coordinates together. This command works in exactly the same way, but when using `to` we can add options to the connecting command. For the green line we give two options, firstly the angle (in degrees) at which the line departs from the first coordinate, and secondly the angle at which the line arrives at the second coordinate. The blue line is a so called Bézier curve, this is a line defined by its begin and end points together with a number of support points (in this case one) that in effect draw the line toward themselves. In Ti*k*Z these types of lines can be defined with either one or two support points. In the example below we give the code to draw a Bézier curve with two support points.

```
\begin{tikzpicture}
  \draw[dotted,gray] (-1,0) -- (5,1);
  \draw (-1,0) .. controls (0,-2) and (4,2) .. (5,1);
\end{tikzpicture}
```

The result of this code is the following image.

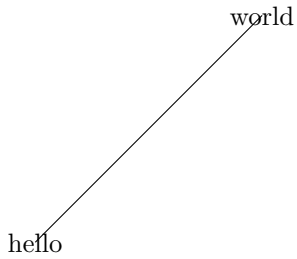


8.1.3 Nodes

The next step on the way to typesetting a graph are the labels, the bits of text and numbers that are contained in the figure. To write text in a Ti*k*Z figure we need an object called a *node*. The simplest example of a node is some text which is placed on a given coordinate, but a node can also be a bit of text surrounded by a circle, rectangle, or other simple shape. A node does not necessarily need to contain text, but for most cases text is the purpose of the node. A node can be defined while drawing a path, which works as follows:

```
\begin{tikzpicture}[scale=3]
  \draw (0,0) node {hello} -- (1,1) node {world};
\end{tikzpicture}
```

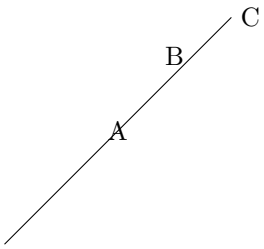
The result of this code is the following image.



The option `[scale=3]` at the beginning of the figure increases the scale of the figure with a factor 3. The command node can also be given options, for example to place a node in the middle of a line, at the end of a line, etc.

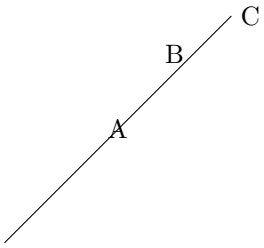
```
\begin{tikzpicture}[scale=3]
  \draw (0,0) -- (1,1) node[midway]{A} node[pos=0.75,above]{B}
   ↪ node[right]{C};
\end{tikzpicture}
```

The result of this code is the following image.



Please note: when using the command `to` rather than `--`, the location of the node command changes slightly. This following example code generates the same image as given above.

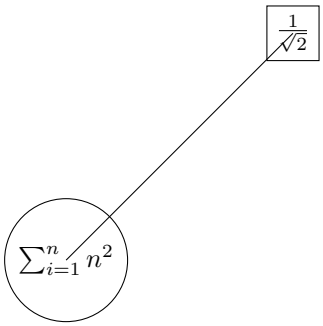
```
\begin{tikzpicture}[scale=3]
  \draw (0,0) to node[midway]{A} node[pos=0.75, above]{B}
   ↪  (1,1) node[right]{C};
\end{tikzpicture}
```



Nodes can also be given mathematical formulae, and as has been said they can be bordered by a simple shape.

```
\begin{tikzpicture}[scale=3]
  \draw (0,0) node[circle, draw]{$\sum_{i=1}^{n}n^2$} -- (1,1)
  node[rectangle,draw]{$\frac{1}{\sqrt{2}}$};
\end{tikzpicture}
```

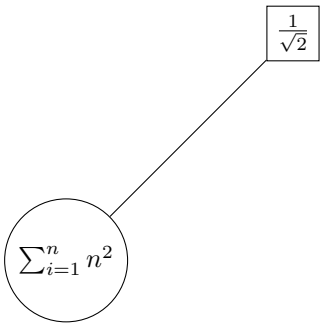
The result of this code is the following image.



As you might have spotted, the line connecting the nodes still starts and ends exactly in the defined coordinates, which is not very pretty. This can be solved by predefining the nodes, and giving them a label, so that we can draw the line from node to node.

```
\begin{tikzpicture}[scale=3]
  % define nodes
  \node[circle,draw] (label1) at (0,0) {$\sum_{i=1}^{n}n^2$};
  \node[rectangle,draw] (label2) at (1,1)
   ↪  {$\frac{1}{\sqrt{2}}$};

  % draw the line
  \draw (label1) -- (label2);
\end{tikzpicture}
```

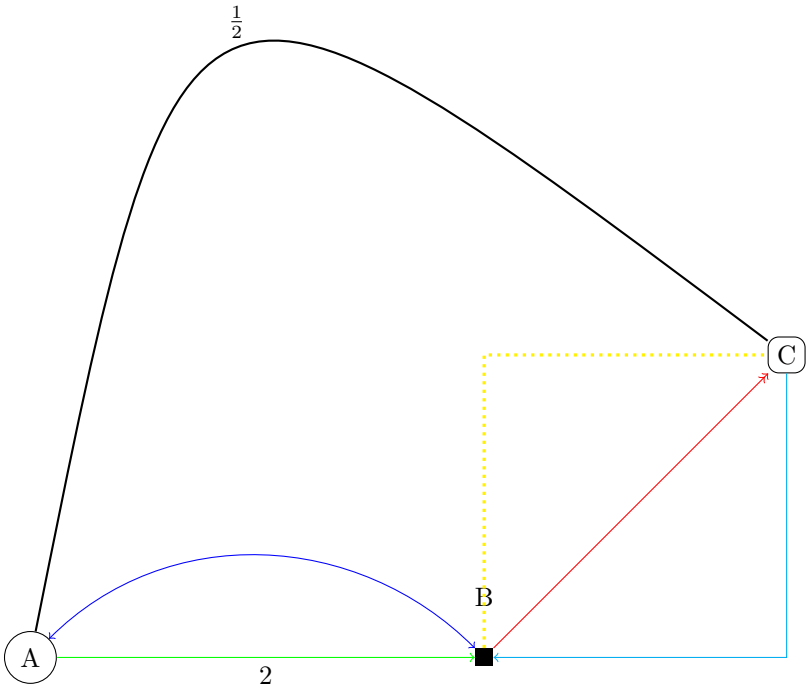
The result of this code is the following image.



We have now seen sufficient examples to understand how a graph can be typeset in Ti*k*Z. The next bit of code is the code used to generate the graph seen at the beginning of this lesson.

```
\begin{tikzpicture}[scale=2]
```

```
  % Define the nodes
  \node[circle, draw] at (0,0) (a) {A};
  \node[rectangle, fill] at (3,0) (b) {};
  \node at (3,0.4) (blabel) {B};
  \node[rectangle,rounded corners, draw] at (5,2) (c) {C};

  % Draw the paths
  \draw[->, green] (a) -- (b) node[midway, below,black]{2};
  \draw[<->, blue] (a) to[out=45, in=135] (b);
  \draw[-»,red] (b)--(c);
  \draw[yellow,dotted,very thick] (b) |- (c);
  \draw[<-,cyan] (b) -| (c);
  \draw[thick,black] (a).. controls (1,5) .. (c) node[midway,
   ↪  above]{$\frac{1}{2}$};
\end{tikzpicture}
```

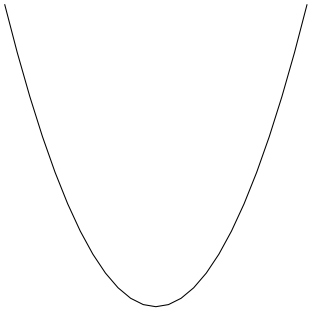


8.1.4 Plotting curves

It is not only possible to draw lines between coordinates in Ti*k*Z. There is also a built in way to plot curves of a large range of different functions. For example the function $y = x^2$ on the interval $[-2, 2]$.

```
\begin{tikzpicture}
  \draw [domain=-2:2] plot (\x, {pow(\x,2}});
\end{tikzpicture}
```

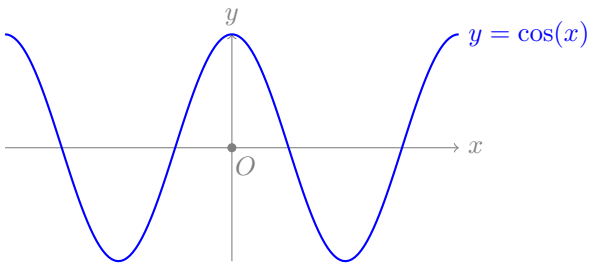
The result of this code is the following image.



By using what you have learned in the previous sections, you can add axes and labels to this curve. For example:

```
\begin{tikzpicture}[scale=1.5]
  % Draw the x and y axis, label the axes and the origin
  \draw[gray, ->] (-2,0) -- (2,0) node[right]{$x$}
   ↪  node[pos=0.53, below]{$O$};
  \draw[gray, ->] (0,-1) -- (0,1) node[above]{$y$};
  \draw[fill,gray] (0,0) circle [radius=1pt];

  % Plot the curve
  \draw[blue, thick] [domain=-2:2, samples=150] plot (\x,
   ↪  {cos(pi*\x r)})
  node[right]{$y = \cos(x)$};
  % Note: the r in the argument of the cosine signifies that
   ↪  we enter \x in radians
\end{tikzpicture}
```

The result of this code is the following image.

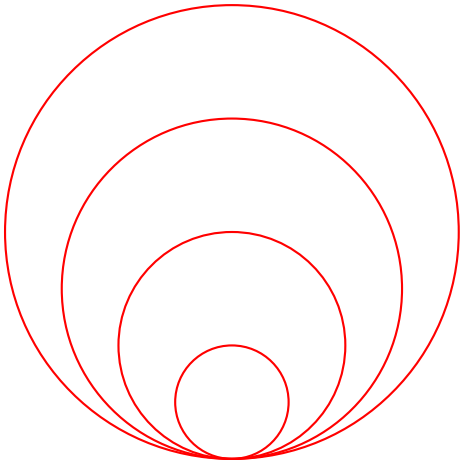


8.1.5 Working with loops

In Ti*k*Z it is also possible to make use of for loops, using `\foreach ... in ...`. In this way, a fairly complicated image can be typeset very quickly.

```
\begin{tikzpicture}[scale=0.75]
  \foreach \x in {0,1,2,3}
  \draw[red,thick] (0,\x) circle [radius=\x+1];
\end{tikzpicture}
```

The result of this code is the following image.

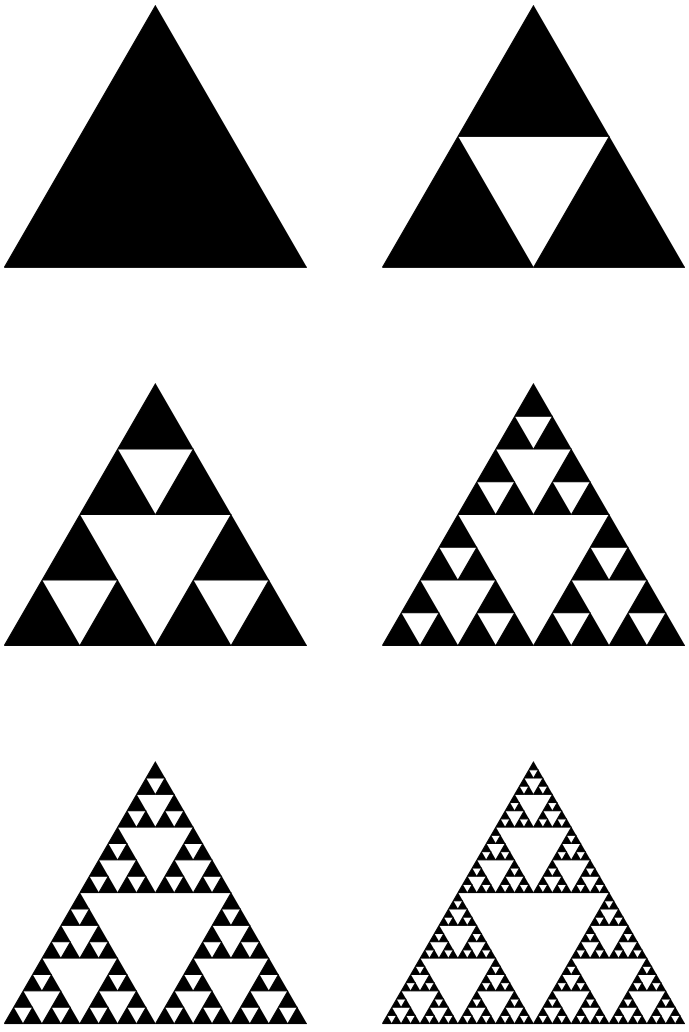


This idea is very useful for drawing iterative figures like Sierpiński's triangle, although that does require a special Ti*k*Z library with which a recursive function can be defined. The following code gives the first six iterations of Sierpiński's triangle, drawn in Ti*k*Z. The code also contains some things in the preamble (before `\begin{document}`), so we give the full LaTeX code for a pdf with only this figure in it.

```
\documentclass[border=1cm]{standalone}

\usepackage{tikz}
\usetikzlibrary{math}

% Define a equilateral triangle with lower left corner at
 ↪  coordinate #1 and
% with length of the sides #2
\newcommand\Triangle[2]{
  \draw #1 coordinate(a)  -- ++(0:#2) coordinate(b) ;
  \draw (a) -- ++(60:#2) coordinate(c);
  \fill (a) -- (b) -- (c) -- cycle;
}

\begin{document}
\begin{tikzpicture}
  \tikzmath{
    % Define the recursive function sierpinski
    function sierpinski(\x, \y, \s, \d) {
      if (\d == 0) then {
        % Draw a triangle lower left corner at (\x, \y),
         ↪  length \s
        { \Triangle{(\x,\y)}{\s}; };
            } else {
        % Rescale the length of the sides and choose correct
         ↪  coords
    % for the next triangles
        \u1 = 0.25*\s;
        \u2 = \u1*sqrt(3);
        \u3 = 0.5*\s;
        sierpinski(\x,\y,\u3,\d-1);
        sierpinski(\x+\u3,\y,\u3,\d-1);
        sierpinski(\x+\u1,\y+\u2,\u3,\d-1);
      };
    };
    % Let the length of the sides of the base triangle be 4,
     ↪  and generate 6 figures
    \S = 4;
    for \d in {0,...,5}{
      % To situate all plots nicely under and next to each
       ↪  other, define the coords
      % of the lower left corners preemptively
      \x = (\S+1)*mod(\d,2);
```

```
        \y = int(\d/2) * (\S+1);
        sierpinski(\x,-\y,\S,\d);
        };
  }
\end{tikzpicture}
\end{document}
```

The result of this code is the following image.



8.2 Exercises

8.2.1 Exercise 1

It is time to try some typesetting yourself. Open your favourite TeX editor and try to typset graph below this exercise: The graph does not need to look exactly the same as the image, the point is that you are able to typeset a graph with similar properties. Hint: It might help to define the nodes in a circle using polar coordinates. You may copy this code for the document structure, so that you only need to focus on TikZ.

```
\documentclass[border=1cm]{standalone}
\usepackage{tikz}
\begin{document}
\end{document}
```

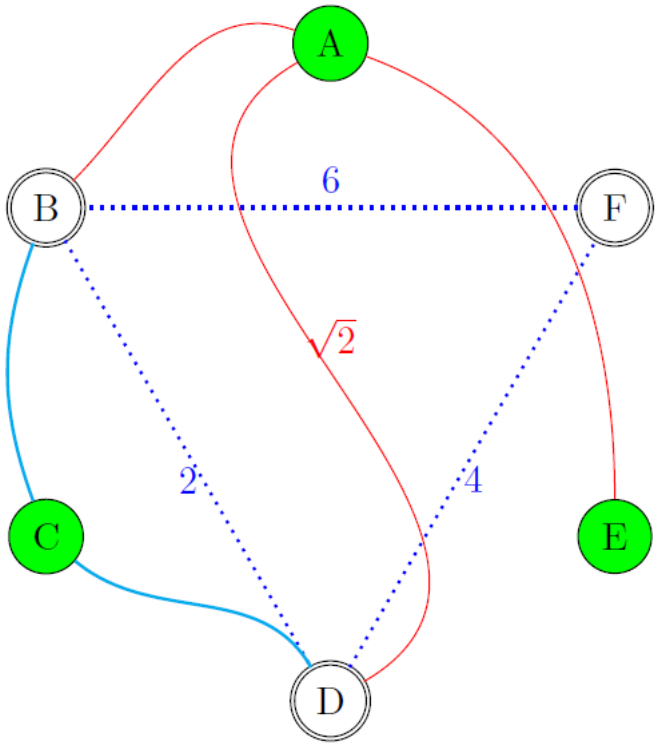


Figure 8.1: Graph for Exercise 1

8.2.2 Exercise 2

Try to typset the plot following this exercise. You may copy this code for the document structure, so that you only need to focus on Ti*k*Z.

```
\documentclass[border=1cm]{standalone}
\usepackage{tikz}
\begin{document}
...
\end{document}
```

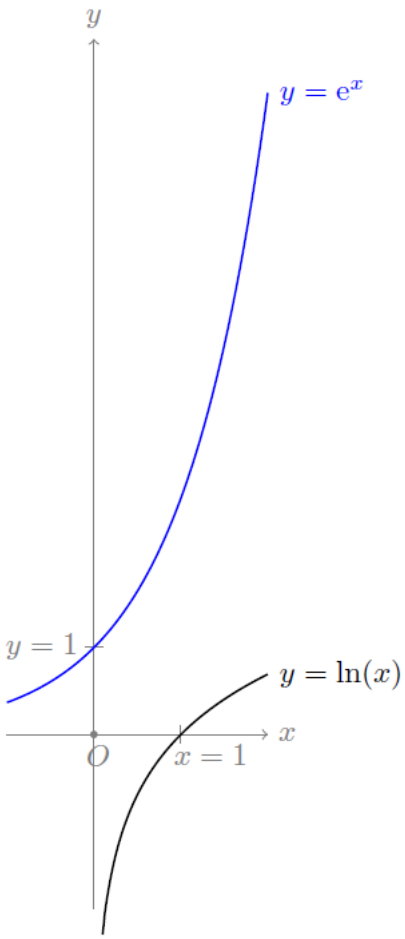


Figure 8.2: TikZ Graph exercise

8.2.3 Exercise 3

Copy the code for the Sierpiński triangles given above, and try to adapt it so that you can generate a figure displaying several iterations of the Sierpiński carpet (see also Wikipedia: https://en.wikipedia.org/wiki/Sierpi%C5%84ski_carpet).