TUTORIAL VIII

TYPESETTING MATHEMATICS

Donal Knuth created TEX primarily to typeset Mathematics beautifully. LATEX includes all the capabilities of TEX in Mathematics typesetting, sometimes with easier user interfaces. Then there are packages like amsmath which enhance and refine these interfaces.

VIII.1. THE BASICS

A mathematical expression occurring in running text (called *in-text* math) is produced by enclosing it between dollar signs. Thus to produce

> The equation representing a straight line in the Cartesian plane is of the form $ax + by + c = 0$, where a, b, c are constants.

we type

```
The equation representing  a straight line in the Cartesian plane
is of the form $ax+by+c=0$, where $a$, $b$, $c$ are constants.
```

Some comments are in order. First note that the text within dollars is typeset in *italic* (actually math italic). Again, even though we did not leave any spaces within `ax+by+c=0`, TEX leaves spaces on either side of the addition signs and the equality sign. On the other hand, even if we type `$ax + by + c = 0$`, the output would be the same: $ax + by + c = 0$. The moral? TEX *has its own spacing rules in math mode*.

To see another instance of this, change the last part of the code above to read

```
... where $a, b, c$ are constants.
```

Saves some typing, does not it? But look at the output.

> The equation representing a straight line in the Cartesian plane is of the form $ax + by + c = 0$, where a, b, c are constants.

Do you see the difference? There are no spaces after the commas, though we had such spaces in the output. So TEX swallows spaces in math mode (you can not save dollars that way!).

Incidentally, dollar signs are TEX way of distinguishing Mathematical text. LATEX has other ways also of doing it, using `\( ... \)` or `\begin{math} ... \end{math}`. Thus either of the inputs shown below also produces the same output as above.

```
The equation representing a straight line in the Cartesian plane is of
the form \(ax+by+c=0\), where \(a\), \(b\), \(c\) are constants.
```

```
The equation representing a straight line in the Cartesian plane is
of the form \begin{math}ax+by+c=0\end{math}, where \begin{math} a
\end{math}, \begin{math} b \end{math}, \begin{math} c \end{math} are
constants.
```

Now suppose we want to *display* the equation in the above output as in

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0$$
>
> where a, b, c are constants.

This can be done by changing the input as follows:

```
The equation representing a straight line in the Cartesian plane is
of the form
$$
ax+by+c=0
$$
where $a$, $b$, $c$ are constants.
```

Again `$$ ... $$` is the TEX way of producing displayed math. LATEX has the constructs `\[ ... \]` or `\begin{displaymath} ... \end{displaymath}` also to do this.

VIII.1.1. Superscripts and subscripts

Look at the text below

> In the seventeenth century, Fermat conjectured that if $n > 2$, then there are no integers x, y, z for which
>
> $$x^n + y^n = z^n.$$
>
> This was proved in 1994 by Andrew Wiles.

This is produced by the input

```
In the seventeenth century, Fermat conjectured that if $n>2$, then
there are no integers $x$, $y$, $z$ for which
$$
x^n+y^n=z^n.
$$
This was proved in 1994 by Andrew Wiles.
```

This shows that superscripts (mathematicians call them exponents) are produced by the ^ symbol. If the superscript is more than one character long, we must be careful to *group* these characters properly. Thus to produce

> It is easily seen that $(x^m)^n = x^{mn}$.

we must type

```
It is easily seen that $(x^m)^n=x^{mn}$.
```

Instead of `$x^{mn}$`, if we type `$x^mn$` we end up with x^mn instead of the intended x^{mn} in the output.

We can have superscripts of superscripts (and mathematicians do need them). For example,

> Numbers of the form $2^{2^n} + 1$, where n is a natural number, are called Fermat numbers.

is produced by

```
Numbers of the form $2^{2^n}+1$, where $n$ is a natural number, are
called Fermat numbers.
```

Note the grouping of superscripts. (What happens if you type `$2^2^n+1$` or `${2^2}^n$`?)

Now let us see how subscripts (mathematicians call them subscripts) are produced. To get

> The sequence (x_n) defined by
>
> $$x_1 = 1, \quad x_2 = 1, \quad x_n = x_{n-1} + x_{n-2} \;\; (n > 2)$$
>
> is called the Fibonacci sequence.

we must type

```
The sequence $(x_n)$ defined by
$$
x_1=1,\quad x_2=1,\quad x_n=x_{n-1}+x_{n-2}\;\;(n>2)
$$
is called the Fibonacci sequence.
```

Thus subscripts are produced by the _ character. Note how we insert spaces by the `\quad` command. (The command `\;` in *math mode* produces what is known as a "thickspace".) Subscripts of subscripts can be produced as in the case of superscripts (with appropriate grouping).

We can also have superscripts and subscripts together. Thus

> If the sequence (x_n) converges to a, then the sequence (x_n^2) converges to a^2

is produced by

```
If the sequence $(x_n)$ converges to $a$, then the sequence
$(x_n^2)$ converges to $a^2$
```

Again, we must be careful about the grouping (or the lack of it) when typesetting superscripts and subscripts together. The following inputs and the corresponding outputs make the point.

```
$$
x_m^n\qquad x^n_m\qquad {x_m}^n\qquad {x^n}_m
$$
```

> $$x_m^n \qquad x^n_m \qquad {x_m}^n \qquad {x^n}_m$$

(This has to do with the way TEX works, producing "boxes" to fit the output characters. The box for x_m^n is like $\boxed{x_m^n}$ while the box for ${x_m}^n$ is $\boxed{x_m}^n$.

VIII.1.2. Roots

Square roots are produced by the `\sqrt` argument. Thus `$\sqrt{2}$` produces $\sqrt{2}$. This command has an optional argument to produce other roots. Thus

> Which is greater $\sqrt[4]{5}$ or $\sqrt[5]{4}$?

is produced by

```
Which is greater $\sqrt[4]{5}$ or $\sqrt[5]{4}$?
```

The horizontal line above the root (called *vinculum* by mathematicians of yore) elongates to accommodate the enclosed text. For example, `$\sqrt{x+y}$` produces $\sqrt{x+y}$. Also, you can produce nested roots as in

The sequence

$$2\sqrt{2},\quad 2^2\sqrt{2-\sqrt{2}},\quad 2^3\sqrt{2-\sqrt{2+\sqrt{2}}},\quad 2^4\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}},\ldots$$

converge to π.

by typing

```
The sequence
 $$
 2\sqrt{2}\,,\quad 2^2\sqrt{2-\sqrt{2}}\,,\quad 2^3
    \sqrt{2-\sqrt{2+\sqrt{2}}}}\,,\quad 2^4\sqrt{2-
     \sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}}\,,\;\ldots
 $$
converge to $\pi$.
```

The `\ldots` command above produces ..., the three dots indicating indefinite continuation, called *ellipsis* (more about them later). The command `\,` produces a "thinspace" (as opposed to a thickspace produced by `\;` , seen earlier). Why all this thin and thick spaces in the above input? Remove them and see the difference. (A tastefully applied thinspace is what makes a mathematical expression typeset in TEX really beautiful.)

The symbol π in the output produced by `$\pi$` maybe familiar from high school mathematics. It is a Greek letter named "pi". Mathematicians often use letters of the Greek alphabet ((which even otherwise is Greek to many) and a multitude of other symbols in their work. A list of available symbols in LATEX is given at the end of this chapter.

VIII.1.3. Mathematical symbols

In the list at the end of this chapter, note that certain symbols are marked to be not available in native LATEX, but only in certain packages. We will discuss some such packages later. Another thing about the list is that they are categorized into classes such as "Binary Relations", "Operators", "Functions" and so on. This is not merely a matter of convenience.

We have noted that TEX leaves some additional spaces around "binary operators" such as + and −. The same is true for any symbol classified as a binary operator. For example, consider the following

For real numbers x and y, define an operation $\circ$ by

$$x \circ y = x + y - xy$$

This operation is associative.

From the list of symbols, we see that $\circ$ is produced by `\circ` and this is classified as a binary operator, so that we can produce this by

```
For real numbers $x$ and $y$, define an operation $\circ$ by
$$
```

```
x\circ y = x+y-xy
$$
This operation is associative.
```

Note the spaces surrounding the ∘ symbol in the output. On the other hand suppose you want

For real numbers x and y, define an operation $\Box$ by

$$x \Box y = x^2 + y^2$$

The list of symbols show that the symbol □ is produced by `\Box` but that it is available only in the package latexsym or amssymb. So if we load one of these using the `\usepackage` command and then type

```
For real numbers $x$ and $y$, define an operation $\Box$ by
$$
x\Box y = x^2+y^2
$$
```

you will only get

For real numbers x and y, define an operation $\Box$ by

$$x\Box y = x^2 + y^2$$

Notice the difference? There are no spaces around □; this is because, this symbol is not by default defined as a binary operator. (Note that it is classified under "Miscellaneous".) But *we* can ask TEX to consider this symbol as a binary operator by the command `\mathbin` before `\Box` as in

```
For real numbers $x$ and $y$, define an operation $\Box$ by
$$
x\mathbin\Box y=x^2+y^2
$$
```

and this will produce the output shown first.

This holds for "Relations" also. TEX leaves some space around "Relation" symbols and we can instruct TEX to consider any symbol as a relation by the command `\mathrel`. Thus we can produce

Define the relation ρ on the set of real numbers by $x \rho y$ iff $x - y$ is a rational number.

by typing

```
Define the relation $\rho$ on the set of real numbers by
$x\mathrel\rho y$ iff $x-y$ is a rational number.
```

(See what happens if you remove the `\mathrel` command.)

VIII.2. CUSTOM COMMANDS

We have seen that LATEX produces mathematics (and many other things as well) by means of "commands". The interesting thing is that we can build our own commands using the ones available. For example, suppose that t the expression $(x_1, x_2, \ldots, x_n)$ occurs frequently in a document. If we now write

```
\newcommand{\vect}{(x_1,x_2,\dots,x_n)}
```

Then we can type `$\vect$` anywhere after wards to produce $(x_1, x_2, \dots, x_n)$ as in

```
We often write $x$ to denote the vector $\vect$.
```

to get

> We often write x to denote the vector $(x_1, x_2, \dots, x_n)$.

(By the way, the best place to keep such "newcommands" is the preamble, so that you can use them anywhere in the document. Also, it will be easier to change the commands, if the need arises).

OK, we can now produce $(x_1, x_2, \dots, x_n)$ with `$\vect$`, but how about $(y_1, y_2, \dots, y_n)$ or $(z_1, z_2, \dots, z_n)$? Do we have to define newcommands for each of these? Not at all. We can also define commands with *variable arguments* also. Thus if we change our definition of `\vect` to

```
\newcommand{\vect}[1]{(#1_1,#1_2,\dots,#1_n)}
```

Then we can use `$\vect{x}$` to produce $(x_1, x_2, \dots, x_n)$ and `$\vect{a}$` to produce $(a_1, a_2, \dots, a_n)$ and so on.

The form of this definition calls for some comments. The `[1]` in the `\newcommand` above indicates that the command is to have *one* (variable) argument. What about the `#1`? Before producing the output, each occurrence of `#1` will be replaced by the (single) argument we supply to `\vect` in the input. For example, the input `$\vect{a}$` will be changed to `$(a_1,a_2,\dots,a_n)$` at some stage of the compilation.

We can also define commands with more than one argument (the maximum number is 9). Thus for example, if the document contains not only $(x_1, x_2, \dots, x_n)$, $(y_1, y_2, \dots, y_n)$ and so on, but $(x_1, x_2, \dots, x_m)$, $(y_1, y_2, \dots, y_p)$ also, then we can change our definition of `\vect` to

```
\newcommand{\vect}[2]{(#1_1,#1_2,\dotsc,#1_#2)}
```

so that we can use `$\vect{x}{n}$` to produce $(x_1, x_2, \dots, x_n)$ and `$\vect{a}{p}$` to produce $(a_1, a_2, \dots, a_p)$.

VIII.3. MORE ON MATHEMATICS

There are some many other features of typesetting math in LaTeX, but these have better implementations in the package amsmath which has some additional features as well. So, for the rest of the chapter the discussion will be with reference to this package and some allied ones. Thus all discussion below is under the assumption that the package amsmath has been loaded with the command `\usepackage{amsmath}`.

VIII.3.1. Single equations

In addition to the LaTeX commands for displaying math as discussed earlier, the amsmath also provides the `\begin{equation*} ... \end{equation*}` construct. Thus with this package loaded, the output

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0$$
>
> where a, b, c are constants.

can also be produced by

```
The equation representing a straight line in the Cartesian plane is
of the form
\begin{equation*}
  ax+by+c=0
\end{equation*}
where $a$, $b$, $c$ are constants.
```

Why the * after `equation`? Suppose we try it without the * as

```
The equation representing a straight line in the Cartesian plane is
of the form
\begin{equation}
  ax+by+c=0
\end{equation}
where $a$, $b$, $c$ are constants.
```

we get

The equation representing a straight line in the Cartesian plane is of the form

$$ax + by + c = 0 \tag{VIII.1}$$

where a, b, c are constants.

This provides the equation with a *number*. We will discuss equation numbering in some more detail later on. For the time being, we just note that for any environment name with a star we discuss here, the unstarred version provides the output with numbers.

Ordinary text can be inserted inside an equation using the `\text` command. Thus we can get

Thus for all real numbers x we have

$$x \le |x| \quad \text{and} \quad x \ge |x|$$

and so

$$x \le |x| \quad \text{for all } x \text{ in } R.$$

from

```
Thus for all real numbers $x$ we have
\begin{equation*}
  x\le|x|\quad\text{and}\quad x\ge|x|
\end{equation*}
and so
\begin{equation*}
  x\le|x|\quad\text{for all $x$ in $R$}.
\end{equation*}
```

Note the use of dollar signs in the second `\text` above to produce mathematical symbols within `\text`.

Sometimes a single equation maybe too long to fit into one line (or sometimes even *two* lines). Look at the one below:

$$(a+b+c+d+e)^2 = a^2+b^2+c^2+d^2+e^2 + 2ab+2ac+2ad+2ae+2bc+2bd+2be+2cd+2ce+2de$$

This is produced by the environment `multline*` (note the spelling carefully—it is *not* `mult i line`), as shown below.

```
\begin{multline*}
  (a+b+c+d+e)^2=a^2+b^2+c^2+d^2+e^2\\
       +2ab+2ac+2ad+2ae+2bc+2bd+2be+2cd+2ce+2de
\end{multline*}
```

`multline` can be used for equations requiring more than two lines, but without tweaking, the results are not very satisfactory. For example, the input

```
\begin{multline*}
  (a+b+c+d+e+f)^2=a^2+b^2+c^2+d^2+e^2+f^2\\
       +2ab+2ac+2ad+2ae+2af\\
       +2bc+2bd+2be+2bf\\
       +2cd+2ce+2cf\\
       +2de+2df\\
       +2ef
\end{multline*}
```

produces

$$(a+b+c+d+e+f)^2 = a^2+b^2+c^2+d^2+e^2+f^2 + 2ab+2ac+2ad+2ae+2af + 2bc+2bd+2be+2bf + 2cd+2ce+2cf + 2de+2df + 2ef$$

By default, the `multline` environment places the first line flush left, the last line flush right (except for some indentation) and the lines in between, centered within the display.

A better way to typeset the above multiline (not multline) equation is as follows.

$$\begin{aligned}(a+b+c+d+e+f)^2 &= a^2+b^2+c^2+d^2+e^2+f^2\\ &\quad +2ab+2ac+2ad+2ae+2af\\ &\quad +2bc+2bd+2be+2bf\\ &\quad +2cd+2ce+2cf\\ &\quad +2de+2df\\ &\quad +2ef\end{aligned}$$

This is done using the `split` environment as shown below.

```
\begin{equation*}
  \begin{split}
    (a+b+c+d+e+f)^2 & = a^2+b^2+c^2+d^2+e^2+f^2\\
                    &\quad +2ab+2ac+2ad+2ae+2af\\
                    &\quad +2bc+2bd+2be+2bf\\
                    &\quad +2cd+2ce+2cf\\
                    &\quad +2de+2df\\
                    &\quad +2ef
  \end{split}
\end{equation*}
```

Some comments seems to be in order. First note that the `split` environment cannot be used independently, but only inside some equation structure such as `equation` (and others we will soon see). Unlike `multline`, the `split` environment provides for alignment among the "split" lines (using the & character, as in `tabular`). Thus in the above example, all the + signs are aligned and these in turn are aligned with a point a `\quad` to the right of the = sign. It is also useful when the equation contains multiple equalities as in

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2+ab+ba+b^2\\ &= a^2+2ab+b^2\end{aligned}$$

which is produced by

```
\begin{equation*}
  \begin{split}
    (a+b)^2 & = (a+b)(a+b)\\
            & = a^2+ab+ba+b^2\\
            & = a^2+2ab+b^2
  \end{split}
\end{equation*}
```

VIII.3.2. Groups of equations

A group of displayed equations can be typeset in a single go using the `gather` environment. For example,

$$\begin{aligned}(a,b)+(c,d) &= (a+c,b+d)\\ (a,b)(c,d) &= (ac-bd,ad+bc)\end{aligned}$$

can be produced by

```
\begin{gather*}
  (a,b)+(c,d)=(a+c,b+d)\\
  (a,b)(c,d)=(ac-bd,ad+bc)
\end{gather*}
```

Now when several equations are to be considered one unit, the logically correct way of typesetting them is with some alignment (and it is perhaps easier on the eye too). For example,

Thus x, y and z satisfy the equations

$$\begin{aligned}x+y-z &= 1\\ x-y+z &= 1\end{aligned}$$

This is obtained by using the `align*` environment as shown below

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1
\end{align*}
```

We can add a short piece of text between the equations, without disturbing the alignment, using the `\intertext` command. For example, the output

Thus x, y and z satisfy the equations

$$x+y-z = 1$$
$$x-y+z = 1$$

and by hypothesis

$$x+y+z = 1$$

is produced by

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1\\
  \intertext{and by hypothesis}
  x+y+z & =1
\end{align*}
```

We can also set multiple 'columns' of aligned equations side by side as in

Compare the following sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \qquad \cosh^2 x + \sinh^2 x = \cosh 2x$$

All that it needs are extra `&`'s to separate the columns as can be sen from the input

```
Compare the following sets of equations
\begin{align*}
  \cos^2x+\sin^2x & = 1        & \cosh^2x-\sinh^2x & = 1\\
  \cos^2x-\sin^2x & = \cos 2x & \cosh^2x+\sinh^2x & = \cosh 2x
\end{align*}
```

We can also adjust the horizontal space between the equation columns. For example,

```
Compare the sets of equations
\begin{align*}
  \cos^2x+\sin^2x & = 1        &\qquad \cosh^2x-\sinh^2x & = 1\\
  \cos^2x-\sin^2x & = \cos 2x &\qquad \cosh^2x+\sinh^2x & = \cosh 2x
\end{align*}
```

gives

Compare the sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \qquad \cosh^2 x + \sinh^2 x = \cosh 2x$$

Perhaps a nicer way of typesetting the above is

Compare the following sets of equations

$$\begin{aligned}\cos^2 x + sin^2 x &= 1 \\ \cos^2 x - \sin^2 x &= \cos 2x\end{aligned} \quad \text{and} \quad \begin{aligned}\cosh^2 x - \sinh^2 x &= 1 \\ \cosh^2 x + \sinh^2 x &= \cosh 2x\end{aligned}$$

This cannot be produced by the equation structures discussed so far, because any of these environments takes up the entire width of the text for its display, so that we cannot put anything else on the same line. So amsmath provides variants `gathered`, `aligned` and `alignedat` which take up only the *actual width of the contents* for their display. Thus the above example is produced by the input

```
Compare the following sets of equations
\begin{equation*}
  \begin{aligned}
    \cos^2x+sin^2x  & = 1\\
    \cos^2x-\sin^2x & = \cos 2x
  \end{aligned}
  \qquad\text{and}\qquad
  \begin{aligned}
    \cosh^2x-\sinh^2x & = 1\\
    \cosh^2x+\sinh^2x & = \cosh 2x
  \end{aligned}
\end{equation*}
```

Another often recurring structure in mathematics is a display like this

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

There is a special environment `cases` in amsmath to take care of these. The above example is in fact produced by

```
\begin{equation*}
  |x| =
  \begin{cases}
    x & \text{if $x\ge 0$}\\
    -x & \text{if $x\le 0$}
  \end{cases}
\end{equation*}
```

VIII.3.3. Numbered equations

We have mentioned that each of the the 'starred' equation environments has a corresponding unstarred version, which also produces numbers for their displays. Thus our very first example of displayed equations with `equation` instead of `equation*` as in

```
The equation representing a straight line in the Cartesian plane is
of the form
\begin{equation}
  ax+by+c=0
\end{equation}
where $a$, $b$, $c$ are constants.
```

produces

The equation representing a straight line in the Cartesian plane is of the form

$$ax + by + c = 0 \tag{VIII.2}$$

where a, b, c are constants.

Why VIII.2 for the equation number? Well, this is Equation number 2 of Chapter VIII, isn't it? If you want the section number also in the equation number, just give the command

```
\numberwithin{equation}{section}
```

We can also override the number LaTeX produces with one of our own design with the `\tag` command as in

```
The equation representing a straight line in the Cartesian plane is
of the form
 \begin{equation}
  ax+by+c=0\tag{L}
\end{equation}
where $a$, $b$, $c$ are constants.
```

which gives

The equation representing a straight line in the Cartesian plane is of the form

$$ax + by + c = 0 \tag{L}$$

where a, b, c are constants.

There is also a `\tag*` command which typesets the *equation label* without parentheses.

What about numbering alignment structures? Except for `split` and `aligned`, all other alignment structures have unstarred forms which attach numbers to *each* aligned equation. For example,

```
\begin{align}
  x+y-z &= 1\\
  x-y+z &= 1
\end{align}
```

gives

$$x + y - z = 1 \tag{VIII.3}$$
$$x - y + z = 1 \tag{VIII.4}$$

Here is also, you can give a label of your own to *any* of the equations with the `\tag` command. Be careful to give the `\tag` *before* the end of line character `\\` though. (See what happens if you give a `\tag` command *after* a `\\`.) You can also suppress the label for any equation with the `\notag` command. These are illustrated in the sample input below:

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
```

```
    x+y-z & = 1\ntag\\
    x-y+z & = 1\notag\\
    \intertext{and by hypothesis}
    x+y+z & =1\tag{H}
  \end{align*}
```

which gives the following output

Thus x, y and z satisfy the equations

$$\begin{aligned} x+y-z &= 1 \\ x-y+z &= 1 \end{aligned}$$

and by hypothesis

$$x+y+z = 1 \tag{H}$$

What about `split` and `aligned`? As we have seen, these can be used only within some other equation structure. The numbering or the lack of it is determined by this parent structure. Thus

```
\begin{equation}
  \begin{split}
    (a+b)^2 & = (a+b)(a+b)\\
            & = a^2+ab+ba+b^2\\
            & = a^2+2ab+b^2
  \end{split}
\end{equation}
```

gives

$$\begin{aligned} (a+b)^2 &= (a+b)(a+b) \\ &= a^2+ab+ba+b^2 \\ &= a^2+2ab+b^2 \end{aligned} \tag{VIII.5}$$

VIII.4. MATHEMATICS MISCELLANY

There are more things Mathematics than just equations. Let us look at how LaTeX and in particular, the amsmath package deals with them.

VIII.4.1. Matrices

Matrices are by definition numbers or mathematical expressions arranged in rows and columns. The amsmath has several environments for producing such arrays. For example

The system of equations

$$\begin{aligned} x+y-z &= 1\\ x-y+z &= 1\\ x+y+z &= 1 \end{aligned}$$

can be written in matrix terms as

$$\begin{pmatrix} 1 & 1 & -1\\ 1 & -1 & 1\\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x\\ y\\ z \end{pmatrix} = \begin{pmatrix} 1\\ 1\\ 1 \end{pmatrix}.$$

Here, the matrix $\begin{pmatrix} 1 & 1 & -1\\ 1 & -1 & 1\\ 1 & 1 & 1 \end{pmatrix}$ is invertible.

is produced by

```
The system of equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1\\
  x+y+z & = 1
\end{align*}
can be written in matrix terms as
\begin{equation*}
  \begin{pmatrix}
    1 &  1 & -1\\
    1 & -1 &  1\\
    1 &  1 &  1
  \end{pmatrix}
  \begin{pmatrix}
    x\\
    y\\
    z
  \end{pmatrix}
  =
  \begin{pmatrix}
    1\\
    1\\
    1
  \end{pmatrix}.
\end{equation*}
Here, the matrix
$\begin{pmatrix}
  1 &  1 & -1\\
  1 & -1 &  1\\
  1 &  1 &  1
\end{pmatrix}$
is invertible.
```

Note that the environment `pmatrix` can be used within in-text mathematics or in displayed math. Why the `p`? There is indeed an environment `matrix` (without a `p`) but it

produces an array *without* the enclosing parentheses (try it). If you want the array to be enclosed within *square brackets*, use `bmatrix` instead of `pmatrix`. Thus

Some mathematicians write matrices within parentheses as in $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ while others prefer square brackets as in $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

is produced by

```
Some mathematicians write matrices within parentheses as in
$
\begin{pmatrix}
  a & b\\
  c & d
\end{pmatrix}
$
while others prefer square brackets as in
$
\begin{bmatrix}
  a & b\\
  c & d
\end{bmatrix}
$
```

There is also a `vmatrix` environment, which is usually used for determinants as in

The determinant $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ is defined by

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

which is obtained from the input

```
The determinant
$
\begin{vmatrix}
  a & b\\
  c & d
\end{vmatrix}
$
is defined by
\begin{equation*}
  \begin{vmatrix}
    a & b\\
    c & d
  \end{vmatrix}
  =ad -bc
\end{equation*}
```

There is a variant `Vmatrix` which encloses the array in double lines. Finally, we have a `Bmatrix` environment which produces an array enclosed within braces { }.

A row of dots in a matrix can be produced by the command `\hdotsfour`. it should be used with an argument specifying the number of columns to be spanned. For example, to get

A general $m \times n$ matrix is of the form

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

we type

```
A general $m\times n$ matrix is of the form
\begin{equation*}
  \begin{pmatrix}
    a_{11} & a_{12} & \dots & a_{1n}\\
    a_{21} & a_{22} & \dots & a_{2n}\\
    \hdotsfor{4}\\
    a_{m1} & a_{m2} & \dots & a_{mn}
  \end{pmatrix}
\end{equation*}
```

The command `\hdotsfor` has also an optional argument to specify the spacing of dots. Thus in the above example, if we use `\hdotsfor[2]{4}`, then the space between the dots is doubled as in

A general $m \times n$ matrix is of the form

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

VIII.4.2. Dots

In the above example, we used the command `\dots` to produce a row of three dots. This can be used in other contexts also. For example,

```
Consider a finite sequence $X_1,X_2,\dots$, its sum $X_1+X_2+\dots$
and product $X_1X_2\dots$.
```

gives

Consider a finite sequence $X_1, X_2, \dots$, its sum $X_1 + X_2 + \dots$ and product $X_1 X_2 \dots$.

Here the dots in all the three contexts are along the "baseline" of the text. Isn't it better to typeset this as

Consider a finite sequence $X_1, X_2, \dots$, its sum $X_1 + X_2 + \cdots$ and product $X_1 X_2 \cdots$.

with *raised* dots for addition and multiplication? The above text is typeset by the input

```
Consider a finite sequence $X_1,X_2,\dotsc$, its sum $X_1+X_2+\dotsb$
and product $X_1X_2\dotsm$.
```

Here `\dotsc` stands for dots to be used with commas, `\dotsb` for dots with binary operations (or relations) and `\dotsm` for multiplication dots. There is also a `\dotsi` for dots with integrals as in

$$\int_{A_1}\int_{A_2}\cdots\int_{A_n} f$$

VIII.4.3. Delimiters

How do we produce something like

Since $\left|\begin{smallmatrix} a & h & g \\ h & b & f \\ g & f & c \end{smallmatrix}\right| = 0$, the matrix $\left(\begin{smallmatrix} a & h & g \\ h & b & f \\ g & f & c \end{smallmatrix}\right)$ is not invertible.

Here the 'small' in-text matrices are produced by the environment `smallmatrix`. This environment *does not* provide the enclosing *delimiters* ( ) or — — which we must supply as in

```
$
\left|\begin{smallmatrix}
        a & h & g\\
        h & b & f\\
        g & f & c
      \end{smallmatrix}\right|
=0
$,
the matrix
$
\left(\begin{smallmatrix}
        a & h & g\\
        h & b & f\\
        g & f & c
      \end{smallmatrix}\right)
$
is not invertible.
```

Why the `\left|...\right|` and `\left{...\right?` These commands `\left` and `\right` enlarge the *delimiter* following them to the size of the enclosed material. To see their effect, try typesetting the above example without these commands. The list of symbols at the end of the chapter gives a list of delimiters that are available off the shelf.

One interesting point about the `\left` and `\right` pair is that, though every `\left` should be matched to a `\right`, the *delimiters* to which they apply need not match. In particular we can produce a single large delimiter produced by `\left` or `\right` by matching it with a matching command followed by a period. For example,

$$\left.\begin{aligned} u_x &= v_y \\ u_y &= -v_x \end{aligned}\right\} \quad \text{Cauchy-Riemann Equations}$$

is produced by

```
\begin{equation*}
  \left.
    \begin{aligned}
      u_x &= v_y\\
      u_y &= -v_x
    \end{aligned}
  \right\}
  \quad\text{Cauchy-Riemann Equations}
\end{equation*}
```

There are instances where the delimiters produced by `\left` and `\right` are too small or too large. For example,

```
\begin{equation*}
  (x+y)^2-(x-y)^2=\left((x+y)+(x-y)\right)\left((x+y)-(x-y)\right)=4xy
\end{equation*}
```

gives

$$(x+y)^2-(x-y)^2=((x+y)+(x-y))\,((x+y)-(x-y))=4xy$$

where the parentheses are all of the same size. But it may be better to make the outer ones a little larger to make the nesting visually apparent, as in

$$(x+y)^2-(x-y)^2=\Big((x+y)+(x-y)\Big)\Big((x+y)-(x-y)\Big)=4xy$$

This is produced using the commands `\bigl` and `\bigr` before the outer parentheses as shown below:

```
\begin{equation*}
  (x+y)^2-(x-y)^2=\bigl((x+y)+(x-y)\bigr)\bigl((x+y)-(x-y)\bigr)=4xy
\end{equation*}
```

Apart from `\bigl` and `\bigr` there are `\Bigl`, `\biggl` and `\Biggl` commands (and their `r` counterparts) which (in order) produce delimiters of increasing size. (Experiment with them to get a feel for their sizes.)

As another example, look at

For n-tuples of complex numbers $(x_1,x_2,\dots,x_n)$ and $(y_1,y_2,\dots,y_n)$ of complex numbers

$$\left(\sum_{k=1}^{n}|x_ky_k|\right)^2\le\left(\sum_{k=1}^{n}|x_k|\right)\left(\sum_{k=1}^{n}|y_k|\right)$$

which is produced by

```
For $n$-tuples of complex numbers $(x_1,x_2,\dotsc,x_n)$ and
$(y_1,y_2,\dotsc,y_n)$ of complex numbers
\begin{equation*}
  \left(\sum_{k=1}^n|x_ky_k|\right)^2\le
        \left(\sum_{k=1}^{n}|x_k|\right)\left(\sum_{k=1}^{n}|y_k|\right)
\end{equation*}
```

Does not the output below look better?

For n-tuples of complex numbers $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$ of complex numbers

$$\Big(\sum_{k=1}^{n} |x_k y_k|\Big)^2 \le \Big(\sum_{k=1}^{n} |x_k|\Big)\Big(\sum_{k=1}^{n} |y_k|\Big)$$

This one is produced by

```
For $n$-tuples of complex numbers $(x_1,x_2,\dotsc,x_n)$ and
$(y_1,y_2,\dotsc,y_n)$ of complex numbers
\begin{equation*}
  \biggl(\sum_{k=1}^n|x_ky_k|\biggr)^2\le
    \biggl(\sum_{k=1}^{n}|x_k|\biggr)\biggl(\sum_{k=1}^{n}|y_k|\biggr)
\end{equation*}
```

Here the trouble is that the delimiters produced by `\left` and `\right` are a bit too large.

VIII.4.4. Putting one over another

Look at the following text

From the binomial theorem, it easily follows that if n is an even number, then

$$1 - \binom{n}{1}\frac{1}{2} + \binom{n}{2}\frac{1}{2^2} - \cdots - \binom{n}{n-1}\frac{1}{2^{n-1}} = 0$$

We have fractions like $\frac{1}{2^{n-1}}$ and *binomial coefficients* like $\binom{n}{2}$ here and the common feature of both is that they have one mathematical expression over another.

Fractions are produced by the `\frac` command which takes two arguments, the numerator followed by the denominator and the binomial coefficients are produced by the `\binom` command which also takes two arguments, the 'top' expression followed by the 'bottom' one. Thus the the input for the above example is

```
From the binomial theorem, it easily follows that if $n$ is an even
number, then
\begin{equation*}
  1-\binom{n}{1}\frac{1}{2}+\binom{n}{2}\frac{1}{2^2}-\dotsb
   -\binom{n}{n-1}\frac{1}{2^{n-1}}=0
\end{equation*}
```

You can see from the first paragraph above that the *size* of the outputs of `\frac` and `\binom` are smaller in text than in display. This default behavior has to be modified sometimes for nicer looking output. For example, consider the following output

Since (x_n) converges to 0, there exists a positive integer p such that

$$|x_n| < \frac{1}{2} \quad \text{for all } n \ge p$$

Would not it be nicer to make the fraction smaller and typeset this as

Since (x_n) converges to 0, there exists a positive integer p such that

$$|x_n| < \tfrac{1}{2} \quad \text{for all } n \ge p$$

The second output is produced by the input

```
Since $(x_n)$ converges to $0$, there exists a positive integer $p$
such that
\begin{equation*}
  |x_n|<\tfrac{1}{2}\quad\text{for all $n\ge p$}
\end{equation*}
```

Note the use of the command `\tfrac` to produce a smaller fraction. (The first output is produced by the usual `\frac` command.)

There is also command `\dfrac` to produce a display style (larger size) fraction in text. Thus the sentence after the first example in this (sub)section can be typeset as

We have fractions like $\dfrac{1}{2^{n-1}}$ and ...

by the input

```
We have fractions like $\dfrac{1}{2^{n-1}}$ and ...
```

As can be guessed, the original output was produced by `\frac`. Similarly, there are commands `\dbinom` (to produce display style binomial coefficients) and `\tbinom` (to produce text style binomial coefficients).

There is also a `\genfrac` command which can be used to produce custom fractions. To use it, we will have to specify six things

1. The left delimiter to be used—note that { must be specified as `\{`
2. The right delimiter—again, } to be specified as `\}`
3. The thickness of the horizontal line between the top expression and the bottom expression. If it is not specified, then it defaults to the 'normal' thickness. If it is set as `0pt` then there will be no such line at all in the output.
4. The size of the output—this is specified as an integer 0, 1, 2 or 3, greater values corresponding to *smaller* sizes. (Technically these values correspond to `\displaystyle`, `\textstyle`, `\scriptstyle` and `\scriptscriptstyle`.)
5. The top expression
6. The bottom expression

Thus instead of `\tfrac{1}{2}` we can also use `\genfrac{}{}{}{1}{1}{2}` and instead of `\dbinom{n}{r}`, we can also use `\genfrac{(}{)}{0pt}{0}{1}{2}` (but there is hardly any reason for doing so). More seriously, suppose we want to produce $\{{ij \atop k}\}$ and $[{ij \atop k}]$ as in

The Christoffel symbol $\{{ij \atop k}\}$ of the second kind is related to the Christoffel symbol $[{ij \atop k}]$ of the first kind by the equation

$$\left\{{ij \atop k}\right\} = g^{k1}\left[{ij \atop 1}\right] + g^{k2}\left[{ij \atop 2}\right]$$

This can be done by the input

```
The Christoffel symbol $\{{ij \atop k}\}$ of the second kind is related to the Christoffel symbol $[{ij \atop k}]$ of the first kind by the equation
$$\left\{{ij \atop k}\right\} = g^{k1}\left[{ij \atop 1}\right] + g^{k2}\left[{ij \atop 2}\right]$$
```

If such expressions are frequent in the document, it would be better to define 'newcommands' for them and use them instead of `\genfrac` every time as in the following input (which produces the same output as above).

```
\newcommand{\chsfk}[2]{\genfrac{[}{]}{0pt}{}{#1}{#2}}
\newcommand{\chssk}[2]{\genfrac{\{}{\}}{0pt}{}{#1}{#2}}
The Christoffel symbol $\genfrac{\{}{\}}{0pt}{}{ij}{k}$ of the second
kind is related to the Christoffel symbol $\genfrac{[}{]}{0pt}{}{ij}{k}$
of the first kind by the equation
\begin{equation*}
  \chssk{ij}{k}=g^{k1}\chsfk{ij}{1}+g^{k2}\chsfk{ij}{2}
\end{equation*}
```

While on the topic of fractions, we should also mention the `\cfrac` command used to typeset continued fractions. For example, to get

$$\frac{4}{\pi} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cdots}}}$$

simply type

```
\begin{equation*}
  \frac{4}{\pi}=1+\cfrac{1^2}{2+
                  \cfrac{3^2}{2+
                   \cfrac{5^2}{2+\dotsb}}}
\end{equation*}
```

Some mathematicians would like to write the above equation as

$$\frac{4}{\pi} = 1 + \frac{1^2}{2} + \frac{3^2}{2} + \frac{5^2}{2} + \cdots$$

There is no ready-to-use command to produce this, but we can define one as follows

```
\newcommand{\cfplus}{\mathbin{\genfrac{}{}{0pt}{}{}{+}}}
\begin{equation*}
  \frac{4}{\pi}
    =1+\frac{1^2}{2}\cfplus\frac{3^2}{2}\cfplus\frac{5^2}{2}\cfplus\dotsb
\end{equation*}
```

VIII.4.5. Affixing symbols—over or under

The table at the end of this chapter gives various math mode *accents* such as `$\hat{a}$` to produce $\hat{a}$ and `$\dot{a}$` to produce $\dot{a}$. But what if one needs $\overset{\circ}{a}$ or $\underset{\circ}{a}$? The commands `\overset` and `\underset` come to the rescue. Thus `$\overset{\circ}{a}$` produces $\overset{\circ}{a}$ and `$\underset{\circ}{a}$` produces $\underset{\circ}{a}$.

Basic LaTeX provides the commands `\overrightarrow` and `\overleftarrow` also to put (extensible) arrows over symbols, as can be seen from the table. The amsmath package also provides the commands `\underrightarrow` and `\underleftarrow` to put (extensible) arrows *below* mathematical expressions.

Speaking of arrows, amsmath provides the commands `\xrightarrow` and `\xleftarrow` which produces arrows which can accommodate long texts as superscripts or subscripts. Thus we can produce

Thus we see that

$$0 \to A \xrightarrow{f} B \xrightarrow{g} C \to 0$$

is a short exact sequence

from the input

```
Thus we see that
\begin{equation*}
  0\xrightarrow{} A\xrightarrow{f}
                  B\xrightarrow{g}
                  C\xrightarrow{} 0
\end{equation*}
is a short exact sequence
```

Note how the *mandatory* arguments of the first and last arrows are left empty to produce arrows with no superscripts. These commands also allow an *optional* argument (to be typed inside *square brackets*), which can be used to produce subscripts. For example

```
Thus we get
\begin{equation*}
  0\xrightarrow{} A\xrightarrow[\text{monic}]{f}
                  B\xrightarrow[\text{epi}]{g}
                  C\xrightarrow{} 0
\end{equation*}
```

gives

Thus we get

$$0 \to A \xrightarrow[\text{monic}]{f} B \xrightarrow[\text{epi}]{g} C \to 0$$

By the way, would not it be nicer to make the two middle arrows the same width? This can be done by changing the command for the third arrow (the one from `B`) as shown below

```
Thus we get
\begin{equation*}
  0\xrightarrow{} A\xrightarrow[\text{monic}]{f}
                  B\xrightarrow[\hspace{7pt}\text{epi}\hspace{7pt}]{g}
                  C\xrightarrow{}0
\end{equation*}
```

This gives

Thus we get

$$0 \to A \xrightarrow[\text{monic}]{f} B \xrightarrow[\text{epi}]{g} C \to 0$$

where the lengths of the two arrows are *almost* the same. There are indeed ways to make the lengths *exactly* the same, but we will talk about it in another chapter.

Mathematical symbols are also attached as *limits* to such *large operators* as sum ($\sum$), product ($\prod$) set union ($\bigcup$), set intersection ($\bigcap$) and so on. The limits are input as subscripts or superscripts, but their *positioning* in the output is different in text and display. For example, the input

```
Euler not only proved that the series
$\sum_{n=1}^\infty\frac{1}{n^2}$ converges, but also that
\begin{equation*}
  \sum_{n=1}^\infty\frac{1}{n^2}=\frac{\pi^2}{6}
\end{equation*}
```

gives the output

Euler not only proved that the series $\sum_{n=1}^\infty \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^\infty \frac{1}{n^2} = \frac{\pi^2}{6}$$

Note that in display, the sum symbol is larger and the limits are put at the bottom and top (instead of at the sides,which is usually the case for subscripts and superscripts). If you want the *same* type of symbol (size, limits and all) in text also, simply change the line

```
$\sum_{n=1}^\infty\frac{1}{n^2}$
```

to

```
$\displaystyle\sum_{n=1}^\infty\frac{1}{n^2}$
```

and you will get

Euler not only proved that the series $\displaystyle\sum_{n=1}^\infty \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^\infty \frac{1}{n^2} = \frac{\pi^2}{6}$$

(Note that this also changes the size of the fraction. What would you do to keep it small?) On the other hand, to make the displayed operator the same as in the text, add the command `\textstyle` before the `\sum` within the equation.

What if you only want to change the *position of the limits* but not the size of the operator in text? Then change the command `$\sum_{n=1}^\infty \frac{1}{n^2}$` to `$\sum_\limits{n=1}^\infty\frac{1}{n^2}$` and this will produce the output given below.

Euler not only proved that the series $\sum\limits_{n=1}^\infty \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^\infty \frac{1}{n^2} = \frac{\pi^2}{6}$$

On the other hand, if you want side-set limits in display type `\nolimits` after the `\sum` within the equation as in

```
Euler not only proved that the series
$\sum_{n=1}^\infty\frac{1}{n^2}$ converges, but also that
\begin{equation*}
  \sum\nolimits_{n=1}^\infty\frac{1}{n^2}=\frac{\pi^2}{6}
\end{equation*}
```

which gives

Euler not only proved that the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

All these are true for other operators classified as "Variable-sized symbols",except integrals. Though the integral symbol in display is larger, the position of the limits in both text and display is on the side as can be seen from the output below

Thus $\lim_{x\to\infty} \int_0^x \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$ and so by definition,

$$\int_0^{\infty} \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$$

which is produced by

```
Thus
$\lim\limits_{x\to\infty}\int_0^x\frac{\sin x}{x}\,\mathrm{d}x
      =\frac{\pi}{2}$
and so by definition,
\begin{equation*}
  \int_0^\infty\frac{\sin x}{x}\,\mathrm{d}x=\frac{\pi}{2}
\end{equation*}
```

If you want the limits to be above and below the integral sign, just add the command `\limits` immediately after the `\int` command. Thus

```
Thus
$\lim\limits_{x\to\infty}\int_0^x\frac{\sin x}{x}\,\mathrm{d}x
      =\frac{\pi}{2}$
and so by definition,
\begin{equation*}
  \int\limits_0^\infty\frac{\sin x}{x}\,\mathrm{d}x=\frac{\pi}{2}
\end{equation*}
```

gives

Thus $\lim_{x\to\infty} \int_0^x \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$ and so by definition,

$$\int\limits_0^{\infty} \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$$

Now how do we typeset something like

$$p_k(x) = \prod_{\substack{i=1 \\ i\neq k}}^{n} \left(\frac{x - t_i}{t_k - t_i}\right)$$

where we have two lines of subscripts for $\prod$? There is a command `\substack` which will do the trick. The above output is obtained from

```
\begin{equation*}
  p_k(x)=\prod_{\substack{i=1\\i\ne k}}^n
              \left(\frac{x-t_i}{t_k-t_i}\right)
\end{equation*}
```

The amsmath package has also a `\sideset` command which can be used to put symbols at any of the four corners of a *large operator*. Thus

`$\sideset{_{ll}^{ul}}{_{lr}^{ur}}\bigcup$` produces $\sideset{_{ll}^{ul}}{_{lr}^{ur}}\bigcup$

`$\sideset{}{'}\sum$` produces $\sideset{}{'}\sum$.

VIII.5. NEW OPERATORS

Mathematical text is usually typeset in italics, and TEX follows this tradition. But certain functions in mathematics such as log, sin, lim and so on are traditionally typeset in roman. This is implemented in TEX by the use of commands like `$\log$`, `$\sin$`, `$\lim$` and so on. The symbols classified as "Log-like symbols" in the table at the end of this chapter shows such functions which are predefined in LATEX.

Having read thus far, it may be no surprise to learn that we can define our own "operator names" which receive this special typographic treatment. This is done by the `\DeclareMathOperator` command. Thus if the operator cl occurs frequently in the document, you can make the declaration

```
\DeclareMathOperator{\cl}{cl}
```

in the preamble and then type `$\cl(A)$` to produce cl(A), for example.

Note that an operator defined like this accommodates subscripts and superscripts in the usual way, that is, at its sides. Thus

```
We denote the closure of $A$ in the subspace $Y$ of $X$ by
$\cl_Y(A)$
```

produces

We denote the closure of A in the subspace Y of X by $\mathrm{cl}_Y(A)$

If we want to define a new operator with subscripts and superscripts placed in the "limits" position below and above, then we should use the starred form of the `\DeclareMathOperator` as shown below

```
\DeclareMathOperator*{\esup}{ess\,sup}

For $f\in L^\infty(R)$, we define
\begin{equation*}
  ||f||_\infty=\esup_{x\in R}|f(x)|
\end{equation*}
```

(Note that the declaration *must* be done in the preamble.) This produces the output

For $f \in L^\infty(R)$, we define

$$\|f\|_\infty = \operatorname*{ess\,sup}_{x\in R} |f(x)|$$

(Why the `\,` command in the definition?)

VIII.6. THE MANY FACES OF MATHEMATICS

We have noted that most mathematics is typeset in italics typeface and some mathematical operators are typeset in an upright fashion. There may be need for additional typefaces as in typesetting vectors in boldface.

LaTeX includes several styles to typeset mathematics as shown in the table below

TYPE STYLE	COMMAND	EXAMPLE	
		INPUT	OUTPUT
italic (default)	`\mathit`	`$x+y=z$`	$x+y=z$
roman	`\mathrm`	`$\mathrm{x+y=z}$`	$\mathrm{x+y=z}$
bold	`\mathbf`	`$\mathbf{x+y=z}$`	$\mathbf{x+y=z}$
sans serif	`\mathsf`	`$\mathsf{x+y=z}$`	$\mathsf{x+y=z}$
typewriter	`\mathtt`	`$\mathtt{x+y=z}$`	$\mathtt{x+y=z}$
calligraphic (upper case only)	`\mathcal`	`$\mathcal{X+Y=Z}$`	$\mathcal{X+Y=Z}$

In addition to these, several other math alphabets are available in various packages (some of which are shown in the list of symbols at the end of this chapter).

Note that the command `\mathbf` produces only *roman* boldface and *not math italic* boldface. Sometimes you may need boldface math italic, for example to typeset vectors. For this, amsmath provides the `\boldsymbol` command. Thus we can get

In this case, we define
$$\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}$$

from the input

```
In this case, we define
\begin{equation*}
  \boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}
\end{equation*}
```

If the document contains several occurrences of such symbols, it is better to make a new definition such as

```
\newcommand{\vect}[1]{\boldsymbol{#1}}
```

and then use `$\vect{a}$` to produce $\boldsymbol{a}$ and `$\vect{b}$` to produce $\boldsymbol{b}$ and so on. the additional advantage of this approach is that if you change your mind later and want vectors to be typeset with arrows above them as $\vec{a}$, then all you need is to change the `\boldsymol` part of the definition of `\vect` to `\overrightarrow` and the change will be effected throughout the document.

Now if we change the input of the above example as

```
In this case, we define
\begin{equation*}
  \boldsymbol{a+b=c}
\end{equation*}
```

then we get the output

In this case, we define

$$\boldsymbol{a + b = c}$$

Note that now the symbols + and = are also in boldface. Thus `\boldsymbol` makes bold every math symbol in its scope (provided the bold version of that symbol is available in the current math font).

There is another reason for tweaking the math fonts. Recently, the International Standards Organization (ISO) has established the recognized typesetting standards in mathematics. Some of the points in it are,

1. Simple variables are represented by italic letters as a, x.
2. Vectors are written in boldface italic as $\boldsymbol{a}$, $\boldsymbol{x}$.
3. Matrices may appear in sans serif as in A, X.
4. The special numbers e, i and the differential operator d are written in *upright roman*.

Point 1 is the default in LaTeX and we have seen how point 2 can be implemented. to fulfill Point 4, it is enough if we define something like

```
\newcommand{\me}{\mathrm{e}}
\newcommand{\mi}{\mathrm{i}}
\newcommand{\diff}{\mathrm{d}}
```

and then use `$\me$` for e and `$\mi$` for i and `$\diff x$` for dx.

Point 3 can be implemented using `\mathsf` but it is a bit difficult (but not impossible) if we need them to be in *italic* also. The solution is to create a new math alphabet, say, `\mathsfsl` by the command

```
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
```

(in the preamble) and use it to define a command `\matr` to typeset matrices in this font by

```
\newcommand{\matr}[1]{\ensuremath{\mathsfsl{#1}}}
```

so that `$\maqtr A$` produces $\mathsfsl{A}$.

VIII.7. AND THAT IS NOT ALL!

We have only briefly discussed the basic techniques of typesetting mathematics using LaTeX and some of the features of the amsmath package which helps us in this task. For more details on this package see the document `amsldoc.dvi` which should be available with your TeX distribution. If you want to produce *really* beautiful mathematical documents, read the Master—"The TeX Book" by Donald Knuth, especially Chapter 18, "Fine Points of Mathematics Typing".

VIII.8. SYMBOLS

Table VIII.1: Greek Letters

α	`\alpha`	θ	`\theta`	o	`o`	τ	`\tau`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	υ	`\upsilon`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	ϕ	`\phi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	φ	`\varphi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	χ	`\chi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ψ	`\psi`

ζ	\zeta	ν	\nu	ς	\varsigma	ω	\omega
η	\eta	ξ	\xi				
Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Table VIII.2: Binary Operation Symbols

$\pm$	\pm	$\cap$	\cap	$\diamond$	\diamond	$\oplus$	\oplus
$\mp$	\mp	$\cup$	\cup	$\triangle$	\bigtriangleup	$\ominus$	\ominus
$\times$	\times	$\uplus$	\uplus	$\triangledown$	\bigtriangledown	$\otimes$	\otimes
$\div$	\div	$\sqcap$	\sqcap	$\triangleleft$	\triangleleft	$\oslash$	\oslash
$*$	\ast	$\sqcup$	\sqcup	$\triangleright$	\triangleright	$\odot$	\odot
$\star$	\star	$\vee$	\vee	$\lhd$*	\lhd*	$\bigcirc$	\bigcirc
$\circ$	\circ	$\wedge$	\wedge	$\rhd$*	\rhd*	$\dagger$	\dagger
$\bullet$	\bullet	$\setminus$	\setminus	$\unlhd$*	\unlhd*	$\ddagger$	\ddagger
$\cdot$	\cdot	$\wr$	\wr	$\unrhd$*	\unrhd*	$\amalg$	\amalg
$+$	+	$-$	-				

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table VIII.3: Relation Symbols

$\leq$	\leq	$\geq$	\geq	$\equiv$	\equiv	$\models$	\models
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim	$\perp$	\perp
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq	$\mid$	\mid
$\ll$	\ll	$\gg$	\gg	$\asymp$	\asymp	$\parallel$	\parallel
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx	$\bowtie$	\bowtie
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong	$\bowtie$	\Join*
$\sqsubset$	\sqsubset*	$\sqsupset$	\sqsupset*	$\neq$	\neq	$\frown$	\smile
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\doteq$	\doteq	$\smile$	\frown
$\in$	\in	$\ni$	\ni	$\propto$	\propto	$=$	=
$\vdash$	\vdash	$\dashv$	\dashv	$<$	<	$>$	>
$:$	:						

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table VIII.4: Punctuation Symbols

,	,	;	;	:	\colon	.	\ldotp	·	\cdotp

Table VIII.5: Arrow Symbols

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow

↩	\hookleftarrow	↪	\hookrightarrow	↘	\searrow
↼	\leftharpoonup	⇀	\rightharpoonup	↙	\swarrow
↽	\leftharpoondown	⇁	\rightharpoondown	↖	\nwarrow
⇌	\rightleftharpoons	⇝	\leadsto*		

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table VIII.6: Miscellaneous Symbols

…	\ldots	⋯	\cdots	⋮	\vdots	⋱	\ddots
ℵ	\aleph	′	\prime	∀	\forall	∞	\infty
ℏ	\hbar	∅	\emptyset	∃	\exists	□	\Box*
ı	\imath	∇	\nabla	¬	\neg	◇	\Diamond*
ȷ	\jmath	√	\surd	♭	\flat	△	\triangle
ℓ	\ell	⊤	\top	♮	\natural	♣	\clubsuit
℘	\wp	⊥	\bot	♯	\sharp	♢	\diamondsuit
ℜ	\Re	∥	\\|	\	\backslash	♡	\heartsuit
ℑ	\Im	∠	\angle	∂	\partial	♠	\spadesuit
℧	\mho*	.	.	|	|		

* Not predefined in LaTeX 2ε. Use one of the packages latexsym, amsfonts or amssymb.

Table VIII.7: Variable-sized Symbols

∑	\sum	⋂	\bigcap	⨀	\bigodot
∏	\prod	⋃	\bigcup	⨂	\bigotimes
∐	\coprod	⨆	\bigsqcup	⨁	\bigoplus
∫	\int	⋁	\bigvee	⨄	\biguplus
∮	\oint	⋀	\bigwedge		

Table VIII.8: Log-like Symbols

\arccos	\cos	\csc	\exp	\ker	\limsup	\min	\sinh
\arcsin	\cosh	\deg	\gcd	\lg	\ln	\Pr	\sup
\arctan	\cot	\det	\hom	\lim	\log	\sec	\tan
\arg	\coth	\dim	\inf	\liminf	\max	\sin	\tanh

Table VIII.9: Delimiters

(	(	)	)	↑	\uparrow	⇑	\Uparrow
[	[	]	]	↓	\downarrow	⇓	\Downarrow
{	\{	}	\}	↕	\updownarrow	⇕	\Updownarrow
⌊	\lfloor	⌋	\rfloor	⌈	\lceil	⌉	\rceil
⟨	\langle	⟩	\rangle	/	/	\	\backslash
|	|	‖	\\|			

Table VIII.10: Large Delimiters

⟅	\rmoustache	⟆	\lmoustache	⟯	\rgroup	⟮	\lgroup
⏐	\arrowvert	⦀	\Arrowvert	⎪	\bracevert		

Table VIII.11: Math Mode Accents

$\hat{a}$	\hat{a}	$\acute{a}$	\acute{a}	$\bar{a}$	\bar{a}	$\dot{a}$	\dot{a}
$\breve{a}$	\breve{a}	$\check{a}$	\check{a}	$\grave{a}$	\grave{a}	$\vec{a}$	\vec{a}
$\ddot{a}$	\ddot{a}	$\tilde{a}$	\tilde{a}				

Table VIII.12: Some other Constructions

$\widetilde{abc}$	\widetilde{abc}	$\widehat{abc}$	\widehat{abc}
$\overleftarrow{abc}$	\overleftarrow{abc}	$\overrightarrow{abc}$	\overrightarrow{abc}
$\overline{abc}$	\overline{abc}	$\underline{abc}$	\underline{abc}
$\overbrace{abc}$	\overbrace{abc}	$\underbrace{abc}$	\underbrace{abc}
$\sqrt{abc}$	\sqrt{abc}	$\sqrt[n]{abc}$	\sqrt[n]{abc}
f'	f'	$\frac{abc}{xyz}$	\frac{abc}{xyz}

Table VIII.13: AMS Delimiters

⌜	\ulcorner	⌝	\urcorner	⌞	\llcorner	⌟	\lrcorner

Table VIII.14: AMS Arrows

⇢	\dashrightarrow	⇠	\dashleftarrow
⇇	\leftleftarrows	⇆	\leftrightarrows
⇚	\Lleftarrow	↞	\twoheadleftarrow
↢	\leftarrowtail	↫	\looparrowleft
⇋	\leftrightharpoons	↶	\curvearrowleft
↺	\circlearrowleft	↰	\Lsh
⇈	\upuparrows	↿	\upharpoonleft
⇃	\downharpoonleft	⊸	\multimap
↭	\leftrightsquigarrow	⇉	\rightrightarrows
⇄	\rightleftarrows	⇉	\rightrightarrows
⇄	\rightleftarrows	↠	\twoheadrightarrow
↣	\rightarrowtail	↬	\looparrowright
⇌	\rightleftharpoons	↷	\curvearrowright
↻	\circlearrowright	↱	\Rsh
⇊	\downdownarrows	↾	\upharpoonright
⇂	\downharpoonright	⇝	\rightsquigarrow

Table VIII.15: AMS Negated Arrows

↚	\nleftarrow	↛	\nrightarrow	⇍	\nLeftarrow
⇏	\nRightarrow	↮	\nleftrightarrow	⇎	\nLeftrightarrow

Table VIII.16: AMS Greek

F	\digamma	$\varkappa$	\varkappa

Table VIII.17: AMS Hebrew

ב	\beth	ד	\daleth
ג	\gimel		

Table VIII.18: AMS Miscellaneous

$\hbar$	\hbar	$\hbar$	\hslash
□	\square	◊	\lozenge
∡	\measuredangle	∄	\nexists
⅁	\Game	𝕜	\Bbbk
▲	\blacktriangle	▼	\blacktriangledown
★	\bigstar	∢	\sphericalangle
／	\diagup	＼	\diagdown
△	\vartriangle	▽	\triangledown
Ⓢ	\circledS	∠	\angle
℧	\mho	ⅎ	\Finv
‵	\backprime	∅	\varnothing
■	\blacksquare	⧫	\blacklozenge
∁	\complement	ð	\eth

Table VIII.19: AMS Binary Operators

∔	\dotplus	∖	\smallsetminus	⋒	\Cap
⊼	\barwedge	⩒	\veebar	⩓	\doublebarwedge
⊠	\boxtimes	⊡	\boxdot	⊞	\boxplus
⋉	\ltimes	⋊	\rtimes	⋋	\leftthreetimes
⋏	\curlywedge	⋎	\curlyvee	⊖	\circleddash
◎	\circledcirc	∙	\centerdot	⊺	\intercal
⋓	\Cup	⊟	\boxminus	⋇	\divideontimes
⋌	\rightthreetimes	⊛	\circledast		

Table VIII.20: AMS Binary Relations

≦	\leqq	⩽	\leqslant	⪕	\eqslantless
≲	\lessapprox	≈	\approxeq	⋖	\lessdot
≶	\lessgtr	⋚	\lesseqgtr	⪋	\lesseqqgtr
≒	\risingdotseq	≓	\fallingdotseq	∽	\backsim
⊆	\subseteqq	⋐	\Subset	⊏	\sqsubset
≼	\curlyeqprec	≾	\precsim	⪷	\precapprox
⊴	\trianglelefteq	⊨	\vDash	⊪	\Vvdash
⌢	\smallfrown	≏	\bumpeq	≎	\Bumpeq
⩾	\geqslant	⪖	\eqslantgtr	≳	\gtrsim
⪀	\gtrdot	⋙	\ggg	≷	\gtrless
⪌	\gtreqqless	≖	\eqcirc	≗	\circeq
~	\thicksim	≈	\thickapprox	⊇	\supseteqq

⊐	\sqsupset	⪾	\succcurlyeq	⋟	\curlyeqsucc
≿	\succapprox	⊳	\vartriangleright	⊵	\trianglerighteq
∣	\shortmid	∥	\shortparallel	≬	\between
∝	\varpropto	◀	\blacktriangleleft	∴	\therefore
▶	\blacktriangleright	∵	\because	≲	\lesssim
⋘	\lll	≒	\doteqdot	�logo	\backsimeq
≼	\preccurlyeq	◁	\vartriangleleft	⌣	\smallsmile
≧	\geqq	≳	\gtrapprox	≷	\gtreqless
≜	\triangleq	⋑	\Supset	≳	\succsim
⊩	\Vdash	⋔	\pitchfork	϶	\backepsilon

Table VIII.21: AMS Negated Binary Relations

≮	\nless	≰	\nleq	⩽̸	\nleqslant
≨	\lneq	≨	\lneqq	≨	\lvertneqq
⋦	\lnapprox	⊀	\nprec	⋠	\npreceq
⋦	\precnapprox	≁	\nsim	∤	\nshortmid
⊬	\nvdash	⊭	\nvDash	⋪	\ntriangleleft
⊈	\nsubseteq	⊊	\subsetneq	⊊	\varsubsetneq
⊊	\varsubsetneqq	≯	\ngtr	≱	\ngeq
≱	\ngeqq	≩	\gneq	≩	\gneqq
≵	\gnsim	⋧	\gnapprox	⊁	\nsucc
⋡	\nsucceq	≿	\succnsim	⋧	\succnapprox
∦	\nshortparallel	∦	\nparallel	⊯	\nvDash
⋫	\ntriangleright	⋭	\ntrianglerighteq	⊉	\nsupseteq
⊋	\supsetneq	⊋	\varsupsetneq	⊋	\supsetneqq
⩽̸	\nleqq	≲	\lnsim	≾	\precnsim
∤	\nmid	⋪	\ntrianglelefteq	⊊	\subsetneqq
⩾̸	\ngeqslant	≨	\gvertneqq	⋡	\nsucceq
≇	\ncong	⊯	\nVDash	⊉	\nsupseteqq
⊋	\varsupsetneqq				

Table VIII.22: Math Alphabets

		Required package
ABCdef	\mathrm{ABCdef}	
ABCdef	\mathit{ABCdef}	
ABCdef	\mathnormal{ABCdef}	
$\mathcal{ABC}$	\mathcal{ABC}	
$\mathcal{ABC}$	\mathcal{ABC}	euscript with option: mathcal
	\mathscr{ABC}	euscript with option: mathcr
$\mathfrak{ABCdef}$	\mathfrak{ABCdef}	eufrak
$\mathbb{ABC}$	\mathbb{ABC}	amsfonts or amssymb
$\mathscr{ABC}$	\mathscr{ABC}	mathrsfs

TUTORIAL IX

TYPESETTING THEOREMS

IX.1. THEOREMS IN LaTeX

In Mathematical documents we often have special statements such as *axioms* (which are nothing but the assumptions made) and *theorems* (which are the conclusions obtained, sometimes known by other names like *propositions* or *lemmas*). These are often typeset in different font to distinguish them from surrounding text and given a name and a number for subsequent reference. Such distinguished statements are now increasingly seen in other subjects also. We use the term *theorem-like statements* for all such statements.

LaTeX provides the declaration `\newtheorem` to define the theorem-like statements needed in a document. This command has two arguments, the first for *the name we assign to the environment* and the second, *the name to be printed with the statement*. Thus if you want

Theorem 1. *The sum of the angles of a triangle is* 180°.

you first specify

```
\newtheorem{thm}{Theorem}
```

and then type

```
\begin{thm}
  The sum of the angles of a triangle is $180^\circ$.
\end{thm}
```

Note that in the command `\newtheorem` the first argument can be any name you fancy, instead of the `thm` given here. Also, it is a good idea to keep all your `\newtheorem` commands together in the preamble.

The `\newtheorem` command has a couple of optional arguments which control the way the corresponding statement is numbered. For example if you want the above theorem to be numbered 1.1 (the first theorem of the first section) rather than a plain 1, then you must specify

```
\newtheorem{thm}{Theorem}[section]
```

in the `\newtheorem` command. Then the same input as above for the theorem produces

Theorem IX.1.1. *The sum of the angles of a triangle is* 180°.

The next **Theorem** will be numbered 1.2, the third **Theorem** in the fourth section will be numbered 4.3 and so on.

The other optional argument of the `\newtheorem` command is useful when you have several different types of theorem-like statements (such as lemmas and corollaries) and you want some of them to share the same numbering sequence. For example if you want

Theorem IX.1.2. *The sum of the angles of a triangle is* 180°.

An immediate consequence of the result is the following

Corollary IX.1.3. *The sum of the angles of a quadrilateral is* 360°.

Then you must specify

```
\newtheorem{cor}[thm]{Corollary}
```

after the specification `\newtheorem{thm}[section]` and then type

```
\begin{thm}
  The sum of the angles of a triangle is $180^\circ$.
\end{thm}
```

An immediate consequence of the result is the following

Corollary IX.1.4. *The sum of the angles of a quadrilateral is* 360°.

The optional argument `thm` in the definition of the `cor` environment specifies that "Corollaries" and "Theorems" are to be numbered in the same sequence.

A theorem-like environment defined using the `\newtheorem` command has also an optional argument which is used to give a *note* about the theorem such as the name of its discoverer or its own common name. For example, to get

Theorem IX.1.5 (Euclid). *The sum of the angles of a triangle is* 180°.

you must type

```
\begin{thm}[Euclid]
  The sum of the angles of a triangle is $180^\circ$.
\end{thm}
```

Note the optional argument `Euclid` after the `\begin{thm}`. This use of [...] for optional notes sometimes lead to unintended results. For example, to get

Theorem IX.1.6. [0, 1] *is a compact subset of* $\mathbb{R}$.

if you type

```
\begin{thm}
  [0,1] is a compact subset of $\mathbb{R}$.
\end{thm}
```

then you get

Theorem IX.1.7 (0,1). *is a compact subset of* $\mathbb{R}$.

Do you see what happened? The string `0,1` within [ ] at the beginning of the theorem is considered an optional note by LaTeX ! The correct way is to type

```
\begin{thm}
  $[0,1]$ is a compact subset of $\mathbb{R}$.
\end{thm}
```

Now all the theorem-like statements produced above have the *same typographical form*—name and number in **boldface** and the body of the statement in *italics*. What if you need something like

THEOREM IX.1.1 (EUCLID). *The sum of the angles of a triangle is* 180°.

Such customization is necessitated not only by the aesthetics of the author but often by the whims of the designers in publishing houses also.

IX.2. DESIGNER THEOREMS—THE AMSTHM PACKAGE

The package amsthm affords a high level of customization in formatting theorem-like statements. Let us first look at the predefined *styles* available in this package.

IX.2.1. Ready made styles

The default style (this is what you get if you do not say anything about the style) is termed `plain` and it is what we have seen so far—name and number in boldface and body in italic. Then there is the `definition` style which gives name and number in boldface and body in roman. And finally there is the `remark` style which gives number and name in italics and body in roman.

For example if you put in the preamble

```
\usepackage{amsthm}
\newtheorem{thm}{Theorem}[section]
\theoremstyle{definition}
\newtheorem{dfn}{Definition}[section]
\theoremstyle{remark}
\newtheorem{note}{Note}[section]
\theoremstyle{plain}
\newtheorem{lem}[thm]{Lemma}
```

and then type somewhere in your document

```
\begin{dfn}
A triangle is the figure formed by joining each pair
of three non collinear points by line segments.
\end{dfn}

\begin{note}
A triangle has three angles.
\end{note}

\begin{thm}
The sum of the angles of a triangle is $180^\circ$.
\end{thm}

\begin{lem}
The sum of any two sides of a triangle is greater than or equal to the third.
\end{lem}
```

then you get

Definition IX.2.1. A triangle is the figure formed by joining each pair of three non collinear points by line segments.

Note IX.2.1. A triangle has three angles. [1]note

Theorem IX.2.1. *The sum of the angles of a triangle is* 180°.

Lemma IX.2.2. *The sum of any two sides of a triangle is greater than or equal to the third.*

Note how the `\theoremstyle` command is used to switch between various styles, especially the last `\theoremstyle{plain}` command. Without it, the previous `\theoremstyle{remark}` will still be in force when `lem` is defined and so "Lemma" will be typeset in the `remark` style.

IX.2.2. Custom made theorems

Now we are ready to roll our own "theorem styles". This is done via the `\newtheoremstyle` command, which allows us to control almost all aspects of typesetting theorem like statements. this command has nine parameters and the general syntax is

```
\newtheoremstyle%
  {name}%
  {abovespace}%
  {belowspace}%
  {bodyfont}%
  {indent}%
  {headfont}%
  {headpunct}%
  {headspace}%
  {custom-head-spec}%
```

The first parameter *name* is the name of the new *style*. Note that it is *not* the name of the *environment* which is to be used later. Thus in the example above `remark` is the name of a new style for typesetting theorem like statements and `note` is the name of the environment subsequently defined to have this style (and `Note` is the name of the statement itself).

The next two parameters determine the vertical space between the theorem and the surrounding text—the *abovespace* is the space from the preceding text and the *belowspace* the space from the following text. You can specify either a rigid length (such as `12pt`) or a rubber length (such as `\baselineskip`) as a value for either of these. Leaving either of these empty sets them to the "usual values" (Technically the `\topsep`).

The fourth parameter *bodyfont* specifies the font to be used for the body of the theorem-like statement. This is to be given as a *declaration* such as `\scshape` or `\bfseries` and *not* as a *command* such as `\textsc` or `\textbf`. If this is left empty, then the main text font of the document is used.

The next four parameters refer to the *theoremhead*—the part of the theorem like statement consisting of the name, number and the optional note. The fifth parameter *indent* specifies the indentation of *theoremhead* from the left margin. If this is empty, then there is no indentation of the *theoremhead* from the left margin. The next parameter specifies the font to be used for the *theoremhead*. The comments about the parameter *bodyfont*, made in the previous paragraph holds for this also. The parameter *headpunct*

(the seventh in our list) is for specifying the *punctuation* after the theoremhead. If you do not want any, you can leave this empty. The last parameter in this category (the last but one in the entire list), namely *headspace*, determines the (horizontal) space to be left between the *theoremhead* and the *theorembody*. If you want only a normal interword space here put a *single blank space* as `{ }` in this place. (Note that it is not the same as leaving this *empty* as in `{}`.) Another option here is to put the command `\newline` here. Then instead of a space, you get a linebreak in the output; that is, the *theoremhead* will be printed in a line by itself and the *theorembody* starts from the next line.

The last parameter *custom-head-spec* is for customizing *theoremheads*. Since it needs some explanation (and since we are definitely in need of some breathing space), let us now look at a few examples using the eight parameters we've already discussed.

It is almost obvious now how the last theorem in Section 1 (see Page 111) was designed. It was generated by

```
\newtheoremstyle{mystyle}{}{}{\slshape}{}{\scshape}{.}{ }{}
\theoremstyle{mystyle}
\newtheorem{mythm}{Theorem}[section]
\begin{mythm}
 The sum of the angles of a triangle is $180^\circ$.
\end{mythm}
```

As another example, consider the following

```
\newtheoremstyle{mynewstyle}{12pt}{12pt}{\itshape}%
  {}{\sffamily}{:}{\newline}{}
\theoremstyle{mynewstyle}
\newtheorem{mynewthm}{Theorem}[section]
\begin{mynewthm}[Euclid]
 The sum of the angles of a triangle is $180^\circ$.
\end{mynewthm}
```

This produces

Theorem IX.2.1 (Euclid):
The sum of the angles of a triangle is 180°.

Do you need anything more? Perhaps yes. Note that *theoremhead* includes the optional note to the theorem also, so that the font of the number and name of the theorem-like statement and that of the optional note are always the same. What if you need something like

Cauchy's Theorem (Third Version). *If G is a simply connected open subset of $\mathbb{C}$, then for every closed rectifiable curve γ in G, we have*

$$\int_\gamma f = 0.$$

It is in such cases, that the last parameter of `\newtheoremstyle` is needed. Using it we can separately customize the name and number of the theorem-like statement and also the optional note. The basic syntax for setting this parameter is

`{`*commands*`#1`*commands*`#2`*commands*`#3}`

where `#1` corresponds to the name of the theorem-like statement, `#2` corresponds to its number and `#3` corresponds to the optional note. We are here actually supplying the replacement text for a command `\thmhead` which has three arguments. It is as if we are defining

```
\renewcommand{\thmhead}[3]{...#1...#2...#3}
```

but without actually typing the `\renewcommand{\thmhead}[3]`. For example the theorem above (Cauchy's Theorem) was produced by

```
\newtheoremstyle{nonum}{}{}{\itshape}{}{\bfseries}{.}{ }{#1 (\mdseries #3)}
\theoremstyle{nonum}
\newtheorem{Cauchy}{Cauchy's Theorem}

\begin{Cauchy}[Third Version]
If $G$ is a simply connected open subset of $\mathbb{C}$, then for every closed
rectifiable curve $\gamma$ in $G$, we have
 \begin{equation*}
   \int_\gamma f=0.
 \end{equation*}
\end{Cauchy}
```

Note that the absence of `#2` in the *custom-head-spec*, suppresses the theorem number and that the *space* after `#1` and the command `(\mdseries#3)` sets the optional note in medium size within parentheses and with a preceding space.

Now if you try to produce

> **Riemann Mapping Theorem.** *Every open simply connected proper subset of* $\mathbb{C}$ *is analytically homeomorphic to the open unit disk in* $\mathbb{C}$.

by typing

```
\theoremstyle{nonum}
\newtheorem{Riemann}{Riemann Mapping THeorem}

\begin{Riemann}Every open simply connected proper subset of $\mathbb{C}$ is analytically
homeomorphic to the open unit disk in $\mathbb{C}$.
\end{Riemann}
```

you will get

> **Riemann Mapping Theorem** (). *Every open simply connected proper subset of* $\mathbb{C}$ *is analytically homeomorphic to the open unit disk in* $\mathbb{C}$.

Do you see what is happened? In the `\theoremstyle{diffnotenonum}`, the parameter controlling the *note* part of the *theoremhead* was defined as `(\mdseries #3)` and in the `\newtheorem{Riemann}`, there is no optional note, so that in the output, you get an empty "note", *enclosed in parantheses* (and also with a preceding space).

To get around these difficulties, you can use the commands `\thmname`, `\thmnumber` and `\thmnote` *within* the {*custom-head-spec*} as

```
{\thmname{commands#1}%
\thmnumber{commands#2}%
\thmnote{commands#3}}
```

Each of these three commands will typeset its argument *if and only if the corresponding argument in the* `\thmhead` *is non empty*. Thus the correct way to get the **Riemann Mapping theorem** in Page 114 is to input

```
\newtheoremstyle{newnonum}{}{}{\itshape}{}{\bfseries}{.}{ }%
    {\thmname{#1}\thmnote{ (\mdseries #3)}}

\theoremstyle{newnonum}
\newtheorem{newRiemann}{Riemann Mapping Theorem}

\begin{newRiemann} Every open simply connected proper subset of $\mathbb{C}$ is
analytically homeomorphic to the open unit disk in $\mathbb{C}$.
\end{newRiemann}
```

Then you can also produce **Cauchy's Theorem** in Page 113 by typing

```
\theoremstyle{newnonum}
\newtheorem{newCauchy}{Cauchy's Theorem}

\begin{newCauchy}[Third Version]If $G$ is a simply connected open subset of
$\mathbb{C}$, then for every closed rectifiable curve $\gamma$ in $G$, we have
\begin{equation*}
 \int_\gamma f=0
\end{equation*}
\end{newCauchy}
```

The output will be exactly the same as that seen in Page 113. Now suppose you want to highlight certain theorems from other sources in your document, such as

> **Axiom 1 in [1].** *Things that are equal to the same thing are equal to one another.*

This can be done as follows:

```
\newtheoremstyle{citing}{}{}{\itshape}{}{\bfseries}{.}{ }{\thmnote{#3}}

\theoremstyle{citing}
\newtheorem{cit}{}

\begin{cit}[Axiom 1 in \cite{eu}]
 Things that are equal to the same thing are equal to one another.
\end{cit}
```

Of course, your *bibliography* should include the citation with *label* `eu`.

IX.2.3. There is more!

There are some more predefined features in amsthm package. In all the different examples we have seen so far, the *theorem number* comes after the *theorem name*. Some prefer to have it the other way round as in

> **IX.2.1 Theorem (Euclid).** *The sum of the angles in a triangle is* 180°.

This effect is produced by the command `\swapnumbers` as shown below:

```
\swapnumbers
\theoremstyle{plain}
\newtheorem{numfirstthm}{Theorem}[section]

\begin{numfirstthm}[Euclid]
The sum of the angles in a triangle is $180^\circ$
\end{numfirstthm}
```

Note that the `\swapnumbers` command is a sort of toggle-switch, so that once it is given, *all subsequent theorem-like statements* will have their numbers first. If you want it the other way for some other theorem, then give `\swapnumbers` again before its definition.

A quick way to suppress *theoremnumbers* is to use the `\newtheorem*` command as in

```
\newtheorem*{numlessthm}{Theorem}[section]

\begin{numlessthm}[Euclid]
The sum of the angles in a triangle is $180^\circ$.
\end{numlessthm}
```

to produce

Euclid. *The sum of the angles in a triangle is* 180°.

Note that this could also be done by leaving out `#2` in the *custom-head-spec* parameter of `\newtheoremstyle`, as seen earlier.

We have been talking only about *theorems* so far, but Mathematicians do not live by theorems alone; they need *proofs*. The amsthm package contains a predefined `proof` environment so that the proof of a theorem-like statement can be enclosed within `\begin{proof} ... \end{proof}` commands as shown below:

```
\begin{thmsec}
The number of primes is infinite.
\end{thmsec}

\begin{proof}
Let $\{p_1,p_2,\dotsc p_k\}$ be a finite set of primes. Define $n=p_1p_2\dotsm
p_k+1$. Then either $n$ itself is a prime or has a prime factor. Now  $n$ is
neither equal to nor is divisible by any of the primes $p_1,p_2,\dotsc p_k$ so
that in either case, we get a prime different from $p_1,p_2,\dotsc p_k$. Thus
no finite set of primes can include all the primes.
\end{proof}
```

to produce the following output

Theorem IX.2.3. *The number of primes is infinite.*

Proof. Let $\{p_1, p_2, \dots p_k\}$ be a finite set of primes. Define $n = p_1 p_2 \cdots p_k + 1$. Then either n itself is a prime or has a prime factor. Now n is neither equal to nor is divisible by any of the primes $p_1, p_2, \dots p_k$ so that in either case, we get a prime different from $p_1, p_2, \dots p_k$. Thus no finite set of primes can include all the primes. □

There is an optional argument to the `proof` environment which can be used to change the *proofhead*. For example,

```
\begin{proof}[\textsc{Proof\,(Euclid)}:]
\begin{proof}
Let $\{p_1,p_2,\dotsc p_k\}$ be a finite set of primes. Define $n=p_1p_2\dotsm
p_k+1$. Then either $n$ itself is a prime or has a prime factor. Now  $n$ is
neither equal to nor is divisible by any of the primes $p_1,p_2,\dotsc p_k$ so
that in either case, we get a prime different from $p_1,p_2,\dotsc p_k$. Thus
no finite set of primes can include all the primes.
\end{proof}
```

produces the following

PROOF (EUCLID): Let $\{p_1, p_2, \dots p_k\}$ be a finite set of primes. Define $n = p_1 p_2 \cdots p_k + 1$. Then either n itself is a prime or has a prime factor. Now n is neither equal to nor is divisible by any of the primes $p_1, p_2, \dots p_k$ so that in either case, we get a prime different from $p_1, p_2 \dots p_k$. Thus no finite set of primes can include all the primes. □

Note that the end of a proof is *automatically* marked with a □ which is defined in the package by the command `\qedsymbol`. If you wish to change it, use `\renewcommand` to redefine the `\qedsymbol`. Thus if you like the original "Halmos symbol" ∎ to mark the ends of your proofs, include

```
\newcommand{\halmos}{\rule{1mm}{2.5mm}}
\renewcommand{\qedsymbol}{\halmos}
```

in the preamble to your document.

Again, the placement of the `\qedsymbol` at the *end* of the last line of the proof is done via the command `\qed`. The default placement may not be very pleasing in some cases as in

Theorem IX.2.4. *The square of the sum of two numbers is equal to the sum of their squares and twice their product.*

Proof. This follows easily from the equation

$$(x + y)^2 = x^2 + y^2 + 2xy$$

□

It would be better if this is typeset as

Theorem IX.2.5. *The square of the sum of two numbers is equal to the sum of their squares and twice their product.*

Proof. This follows easily from the equation

$$(x + y)^2 = x^2 + y^2 + 2xy$$

□

which is achieved by the input shown below:

```
\begin{proof}
This follows easily from the equation
\begin{equation}
(x+y)^2=x^2+y^2+2xy\tag*{\qed}
\end{equation}
\renewcommand{\qed}{}
\end{proof}
```

For this trick to work, you must have loaded the package `amsmath` *without* the `leqno` option. Or, if you prefer

Proof. This follows easily from the equation

$$(x + y)^2 = x^2 + y^2 + 2xy \quad \square$$

Then you can use

```
\begin{proof}
This follows easily from the equation
\begin{equation*}
(x+y)^2=x^2+y^2+2xy\qed
\end{equation*}
\renewcommand{\qed}{}
\end{proof}
```

IX.3. HOUSEKEEPING

It is better to keep all `\newtheoremstyle` commands in the preamble than scattering them all over the document. Better still, you can keep them together with other customization in a personal `.sty` file and load it using the `\usepackage` command in the preamble. Also, within this `.sty` file, you can divide your `\newtheorem` commands into groups and preface each group with the appropriate `\theoremstyle`.

BIBLIOGRAPHY

[1] Euclid, *The Elements*, Greece 300 BC

TUTORIAL X

SEVERAL KINDS OF BOXES

The method of composing pages out of boxes lies at the very heart of TEX and many LATEX constructs are available to take advantage of this method of composition.

A *box* is an object that is treated by TEX as a single character. A box cannot be split and broken across lines or pages. Boxes can be moved up, down, left and right. LATEX has three types of boxes.

LR (left-right) The content of this box are typeset from left to right.

Par (paragraphs) This kind of box can contain several lines, which will be typeset in paragraph mode just like normal text. Paragraphs are put one on top of the other. Their widths are controlled by a user specified value.

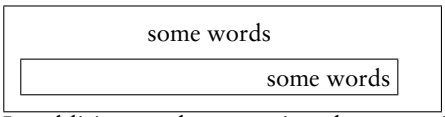
Rule A thin or thick line that is often used to separate various logical elements on the output page, such as between table rows and columns and between running titles and the main text.

X.1. LR BOXES

The usage information of four types of LR boxes are given below. The first line considers the *text* inside the curly braces as a box, with or without a frame drawn around it. For instance, `\fbox{some words}` gives some words whereas `\mbox` will do the same thing, but without the ruled frame around the text.

```
\mbox{text}
\makebox{width}{pos}{text}
\fbox{text}
\framebox{width}{pos}{text}
```

The commands in the third and fourth lines are a generalization of the other commands. They allow the user to specify the width of the box and the positioning of text inside.

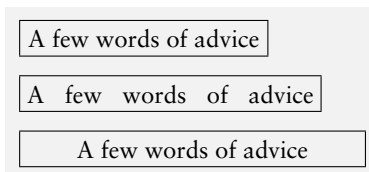



```
\makebox{5cm}{some words}    \par
\framebox{5cm}{r}{some words}
```

In addition to the centering the text with positional argument `[c]` (the default), you can position the text flush left (`[l]`). LATEX also offers you an `[s]` specifier that will stretch your text from the left margin to the right margin of the box provided it contains some stretchable space. The inter-word space is also stretchable and shrinkable to a certain extent.

With LATEX, the above box commands with arguments for specifying the dimensions of the box allow you to make use of four special length parameters: `\width`, `\height`,

\depth and \totalheight. They specify the natural size of the text, where \totalheight is the sum of the \height and \depth.




```
\framebox{A few words of advice}\\[6pt]
\framebox[5cm][s]{A few words of advice}\\[6pt]
\framebox{1.5\width}{A few words of advice}
```

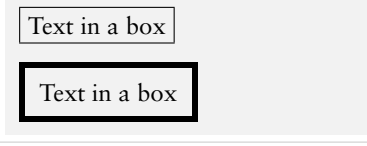
As seen in the margin of the current line, boxes with zero width can be used to make text stick out in the margin. This effect was produced by beginning the paragraph as follows:

```
\makebox{0mm}{r}{$\Leftrightarrow$}
  As seen in the margin of the \dots
```

The appearance of frameboxes can be controlled by two style parameters.

\fboxrule The width of the lines comprising the box produced with the command \fbox or \framebox. The default value in all standard classes is 0.4pt.

\fboxsep The space left between the edge of the box and its contents by \fbox or \framebox. The default value in all standard classes is 3pt.

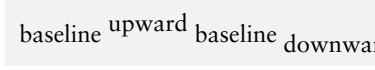



```
\fbox{Text in a box}
\setlength\fboxrule{2pt}\setlength\fboxsep{2mm}
\fbox{Text in a box}
```

Another interesting possibility is to raise or lower boxes. This can be achieved by the very powerful \raisebox command, which has two obligatory and two optional parameters, defined as follows:

\raisebox{*lift*}{*depth*}{*height*}{*contents*}

An example of lowered and elevated text boxes is given below.

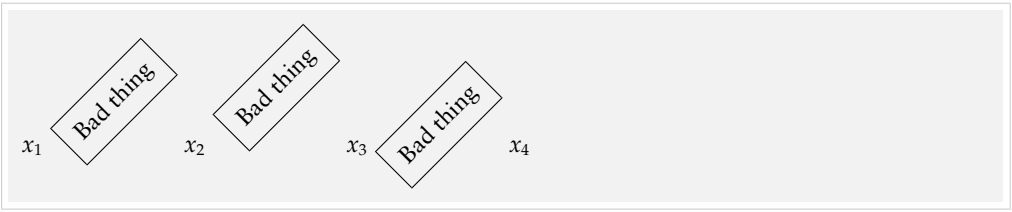



```
baseline \raisebox{1ex}{upward} baseline
      \raisebox{-1ex}{downward} baseline
```

As with \makebox and \framebox the LaTeX implementation of \raisebox offers you the use of the lengths \height, \depth, \totalheight and \width in the first three arguments. Thus, to pretend that a box extends only 90% of its actual height above the baseline you could write:

```
\raisebox{0pt}{0.9\height}{text}
```

or to rotate a box around its lower left corner (instead of its reference point lying on the baseline), you could raise it by its \depth first, e.g.:




```
$x_1$ \doturn{\fbox{Bad thing}}\\
$x_2$ \doturn{\raisebox{\depth}\\
              {\fbox{Bad thing}}}\\
$x_3$ \doturn{\raisebox{-\height}\\
              {\fbox{Bad thing}}} $x_4$
```

X.2. PARAGRAPH BOXES

Paragraph boxes are constructed using the \parbox command or minipage environment. The text material is typeset in paragraph mode inside a box of width *width*. The vertical positioning of the box with respect to the text baseline is controlled by the one-letter optional parameter *pos* ([c], [t], and [b]).

The usage for \parbox command is,

\parbox{*pos*}{*width*}{*text*}

whereas that of the **minipage** environment will be:

\begin{minipage}{*pos*}{*width*}
. . . here goes the text matter . . .
\end{minipage}

The center position is the default as shown by the next example. You can also observe that LaTeX might produce wide inter-word spaces if the measure is incredibly small.

This is the contents of the left-most parbox.

CURRENT LINE

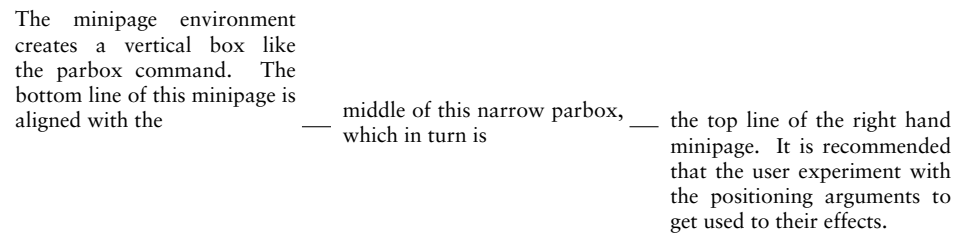
This is the right-most parbox. Note that the typeset text looks sloppy because LaTeX cannot nicely balance the material in these narrow columns.

The code for generating these three \parbox's in a row is given below:

```
\parbox{.3\bs linewidth}
 {This is the contents of the left-most parbox.} \hfill CURRENT LINE \hfill
\parbox{.3\bs linewidth}{This is the right-most parbox. Note that the typeset
 text looks sloppy because \LaTeX{} cannot nicely balance the material in
 these narrow columns.}
```

The **minipage** environment is very useful for the placement of material on the page. In effect, it is a complete mini-version of a page and can contain its own footnotes, paragraphs, and **array**, **tabular** and **multicols** (we will learn about these later) environments. A simple example of minipage environment at work is given below. The baseline is indicated with a small line.

```
\begin{minipage}{b}{.3\linewidth}
 The minipage environment creates a vertical box like the parbox command.
 The bottom line of this minipage is aligned with the
\end{minipage}\hrulefill
\begin{minipage}{c}{.3\linewidth}
 middle of this narrow parbox, which in turn is
\end{minipage}\hrulefill
\begin{minipage}{t}{.3\linewidth}
 the top line of the right hand minipage. It is recommended that the user
 experiment with the positioning arguments to get used to their effects.
\end{minipage}
```



X.3. PARAGRAPH BOXES WITH SPECIFIC HEIGHT

In LaTeX, the syntax of the `\parbox` and **minipage** has been extended to include two more optional arguments.

`\parbox{`*pos*`}{`*height*`}{`*inner pos*`}{`*width*`}{`*text*`}`

is the usage for `\parbox` command, whereas that of the **minipage** environment will be:

`\begin{minipage}{`*pos*`}{`*height*`}{`*inner pos*`}{`*width*`}`
...here goes the text matter ...
`\end{minipage}`

In both cases, *height* is a length specifying the height of the box; the parameters `\height`, `\width`, `\depth`, and `\totalheight` may be employed within the *emph* argument in the same way as in the *width* argument of `\makebox` and `\framebox`.

The optional argument *inner pos* states how the text is to be positioned internally, something that is only meaningful if *height* has been given. Its possible values are:

t To push the text to the top of the box.

b To shove it to the bottom.

c To center it vertically.

s To stretch it to fill up the whole box.

In the last case, we must specify the interline space we wish to have and the deviations allowed from this value as in the example below.

Note the difference between the external positioning argument *pos* and the internal one *inner pos*: the former states how the box is to be aligned with the surrounding text,

while the latter determines how the contents are placed within the box itself. See an example below. We frame the minipages to make it more comprehensible.

This is a minipage with a height of 3 cm with the text aligned at the top.

In this minipage of same height, the text is vertically centered.

In this third box of same height, text is aligned at the bottom.

In this fourth box of same height, the text is stretched to fill in the entire vertical space.

See the code that generated the above boxed material:

```
\begin{minipage}[b][3cm][t]{2cm}
  This is a minipage with a height of 3~cm  with the text aligned
  at the top.
\end{minipage}\hfill
\begin{minipage}[b][3cm][c]{2cm}
  In this minipage of same height, the text is vertically centered.
\end{minipage}}\hfill
\begin{minipage}[b][3cm][b]{2cm}
  In this third box of same height, text is aligned at the bottom.
\end{minipage}\hfill
\begin{minipage}{b}{3cm}{s}{2cm}
  \baselineskip 10pt plus 2pt minus 2pt
  In this fourth box of same height, the  text is stretched to fill in the entire
  vertical space.
\end{minipage}
```

In the last minipage environment the command `\baselineskip` gets the interline space to be 10 points text allows it to be as low as 8 points or as high as 12 points.

X.4. NESTED BOXES

The box commands described above may be nested to any desired level. Including an LR box within a parbox or a minipage causes no obvious conceptual difficulties. The opposite, a parbox within an LR box, is also possible, and is easy to visualize if one keeps in mind that every box is a unit, treated by TEX as a single character of the corresponding size.

A parbox inside an `\fbox` command has the effect that the entire parbox is framed. The present structure was made with

```
\fbox{\fbox{\parbox{.75\linewidth} {A parbox ...}}}
```

This is a parbox of width .75`\linewidth` inside an fbox inside a second fbox, which thus produces the double framing effect.

X.5. RULE BOXES

A rule box is basically a filled-in black rectangle. The syntax for the general command is:

```
\rule{lift}{width}{height}
```

which produces a solid rectangle of width *width* and height *height*, raised above the baseline by an amount *lift*. Thus

```
\rule{8mm}{3mm}
```

generates

and

```
\rule{3in}{.2pt}
```

generates

Without an optional argument *lift*, the rectangle is set on the baseline of the current line of the text. The parameters *lift*, *width* and *height* are all lengths. If *lift* has a negative value, the rectangle is set below the baseline.

It is also possible to have a rule box of zero width. This creates an invisible line with the given *height*. Such a construction is called a *strut* and is used to force a horizontal box to have a desired height or depth that is different from that of its contents.

TUTORIAL XI

FLOATS

XI.1. THE `figure` ENVIRONMENT

Figures are really problematical to present in a document because they never split between pages. This leads to bad page breaks which in turn leave blank space at the bottom of pages. For fine-tuning that document, the typesetter has to adjust the page breaks manually.

But LaTeX provides floating figures which automatically move to suitable locations. So the positioning of figures is the duty of LaTeX.

XI.1.1. Creating floating figures

Floating figures are created by putting commands in a `figure` environment. The contents of the figure environment always remains in one chunk, floating to produce good page breaks. The following commands put the graphic from `figure.eps` inside a floating figure:

```
\begin{figure}
\centering
\includegraphics{figure.eps}
\caption{This is an inserted EPS graphic}
\label{fig1}
\end{figure}
```

Features

- The optional `\label` command can be used with the `\ref`, and `\pageref` commands to reference the caption. The `\label` command must be placed immediately *after* the `\caption`
- If the figure environment contains no `\caption` commands, it produces an unnumbered floating figure.
- If the figure environment contains multiple `\caption` commands, it produces multiple figures which float together. This is useful in constructing side-by-side graphics or complex arrangements.
- A list of figures is generated by the `\listoffigures` command.
- By default, the caption text is used as the caption and also in the list of figures. The caption has an optional argument which specifies the list-of-figure entry. For example,

  `\caption[`*List Text*`]{`*Caption Text*`}`

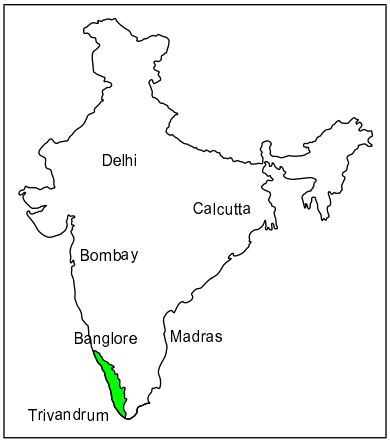


Figure XI.1: This is an inserted EPS graphic

causes "Caption Text" to appear in the caption, but "List Text" to appear in the list of figures. This is useful when using long, descriptive captions.

- The figure environment can only be used in *outer paragraph mode*, preventing it from being used inside any box (such as parbox or minipage).
- Figure environments inside the paragraphs are not processed until the end of the paragraph. For example:

```
..... text text text text text text
  \begin{figure}
  .....
  \end{figure}
..... text text text text text text
```

XI.1.2. Figure placement

The `figure` environment has an optional argument which allows users to specify possible figure locations. The optional argument can contain any combination of the letters: `h`, `t`, `b`, `p`.

h Place the figure in the text where the figure command is located. This option cannot be executed if there is not enough room remaining on the page.
t Place the figure at the top of the page.
b Place the figure at the bottom of a page.
p Place the figure on a page containing only floats.

If no optional arguments are given, the placement options default to `[tbp]`.

When we input a float, LaTeX will read that float and hold it until it can be placed at a better location. Unprocessed floats are those which are read by LaTeX but have not yet been placed on the page. Though the float-placing is done by LaTeX, sometimes the user has to invoke commands to process unprocessed floats. Following commands will do that job:

`\clearpage` This command places unprocessed floats and starts a new page.

`\FloatBarrier` This command causes all unprocessed floats to be processed. This is provided by the `placeins` package. It does not start a new page, unlike `\clearpage`.

Since it is often desirable to keep floats in the section in which they were issued, the `section` option

```
\usepackage[section]{placeins}
```

redefines the `\section` command, inserting a `\FloatBarrier` command before each section. Note that this option is very strict. This option does not allow a float from the previous section to appear at the bottom of the page, since that is after the start of a new section.

The `below` option

```
\usepackage[below]{placeins}
```

is a less-restrictive version of the `section` option. It allows floats to be placed after the beginning of a new section, provided that some of the previous section appears on the page.

`\afterpage/\clearpage` The `afterpage` package provides the `\afterpage` command which executes a command at the next naturally-ocurring page break.

Therefore, using `\afterpage{\clearpage}` causes all unprocessed floats to be cleared at the next page break. `\afterpage{\clearpage}` is especially useful when producing small floatpage figures.

XI.1.3. Customizing float placement

The following style parameters are used by LaTeX to prevent awkward-looking pages which contain too many floats or badly-placed floats.

Float placement counters

`\topnumber` The maximum number of floats allowed at the top of a text page (the default is 2).

`\bottomnumber` The maximum number of floats allowed at the bottom of a text page (the default is 1).

`\totalnumber` The maximum number of floats allowed on any one text page (the default is 3).

These counters prevent LaTeX from placing too many floats on a text page. These counters do not affect float pages. Specifying a `!` in the float placement options causes LaTeX to ignore these parameters. The values of these counters are set with the `\setcounter` command. For example,

```
\setcounter{totalnumber}{2}
```

prevents more than two floats from being placed on any text page.

Figure fractions

The commands given below control what fraction of a page can be covered by floats (where "fraction" refers to the height of the floats divided by `\textheight`). The first

three commands pertain only to text pages, while the last command pertains only to float pages. Specifying a `!` in the float placement options causes LaTeX to ignore the first three parameters, but `\floatpagefraction` is always used. The value of these fractions are set by `\renewcommand`. For example,

```
\renewcommand{\textfraction}{0.3}
```

`\textfraction`	The minimum fraction of a text page which must be occupied by text. The default is 0.2, which prevents floats from covering more than 80% of a text page.
`\topfraction`	The maximum fraction of a text page which can be occupied by floats at the top of the page. The default is 0.7, which prevents any float whose height is greater than 70% of `\textheight` from being placed at the top of a page.
`\bottomfraction`	The maximum fraction of a text page which can be occupied by floats at the bottom of the page. The default is 0.3, which prevents any float whose height is greater than 40% of `\textheight` from being placed at the bottom of a text page.
`\floatpagefraction`	The minimum fraction of a float page that must be occupied by floats. Thus the fraction of blank space on a float page cannot be more than `1-\floatpagefraction`. The default is 0.5.

XI.1.4. Using graphics in LaTeX

This section shows how graphics can be handled in LaTeX documents. While LaTeX can import virtually any graphics format, Encapsulated PostScript (EPS) is the easiest graphics format to import into LaTeX. The 'eps' files are inserted into the file using command `\includegraphics`*file.eps*

The `\includegraphics` command

```
\includegraphics[options]{filename}
```

The following options are available in `\includegraphics` command:

`width`	The width of the graphics (in any of the accepted TeX units).
`height`	The height of the graphics (in any of the accepted TeX units).
`totalheight`	The totalheight of the graphics (in any of the accepted TeX units).
`scale`	Scale factor for the graphic. Specifying `scale = 2` makes the graphic twice as large as its natural size.
`angle`	Specifies the angle of rotation, in degrees, with a counter-clockwise (anti-clockwise) rotation being positive.

Graphics search path

By default, LaTeX looks for graphics files in any directory on the TeX search path. In addition to these directories, LaTeX also looks in any directories specified in the `\graphicspath` command. For example,

```
\graphicspath{{dir1/}{dir2/}}
```

`\includegraphics[`*width=1in*`]{`*tex.png*`}`

`\includegraphics[`*height=1.5in*`]{`*tex.png*`}`

`\includegraphics[`*scale=.25,angle=45*`]{`*tex.png*`}`

`\includegraphics[`*scale=.25,angle=90*`]{`*tex.png*`}`

tells LaTeX to look for graphics files also in `dir1/` and `dir2/`. For Macintosh, this becomes

```
\graphicspath{{dir1:}{dir2:}}
```

Graphics extensions

The `\DeclareGraphicsExtensions` command tells LaTeX which extensions to try if a file with no extension is specified in the `\includegraphics` command. For convenience, a default set of extensions is pre-defined depending on which graphics driver is selected. For example if `dvips` is used, the following graphics extensions (defined in `dvips.def`) are used by default

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

With the above graphics extensions specified, `\includegraphics`*file* first looks for `file.eps`, then `file.ps`, then file `file.eps.gz`, etc. until a file is found. This allows the graphics to be specified with

```
\includegraphics{file}
```

instead of

```
\includegraphics{file.eps}
```

XI.1.5. Rotating and scaling objects

In addition to the `\includegraphics` command, the `graphicx` package includes four other commands which rotate and scale any LaTeX object: text, EPS graphic, etc.

```
\scalebox{2}{\includegraphics{file.eps}}
    \resizebox{4in}{!}{\includegraphics{file.eps}}
    \rotatebox{45}{\includegraphics{file.eps}}
```

produces the same three graphics as

```
\includegraphics[scale=2]{file.eps}
    \includegraphics[width=4in]{file.eps}
    \includegraphics[angle=45]{file.eps}
```

For example, the following are produced with

```
\rotatebox{45}{\fbox{\LARGE{\LaTeX}}}
```

```
\rotatebox{180}{\fbox{\LARGE{\LaTeX}}}
```

However, the `\includegraphics` is preferred because it is faster and produces more efficient PostScript.

XI.2. THE `table` ENVIRONMENT

With the *box* elements already explained in the previous chapter, it would be possible to produce all sorts of framed and unframed tables. However, LaTeXoffers the user far more convenient ways to build such complicated structures.

XI.2.1. Constructing tables

The environments `tabular` and `tabular*` are the basic tools with which tables can be constructed. The syntax for these environments is:

```
\begin{tabular}[pos]{cols} rows \end{tabular}
\begin{tabular*}{width}[pos]{cols} rows    \end{tabular*}
```

Both the above environments actually create a minipage. The meaning of the above arguments is as follows:

pos Vertical positioning arguments (see also the explanation of this argument for parboxes). It can take on the values:

t The top line of the table is aligned with the baseline of the current external line of text.

b The bottom line of the table is aligned with the external baseline.

With no positioning argument given, the table is centered on the external baseline.

width This argument applies only to the `tabular*` environment and determines its overall width. In this case, the *cols* argument must contain the `@`-expression (see below) `@{\extracolsep{\fill}}` somewhere after the first entry. For the other two environments, the total width is fixed by the textual content.

cols The column formatting argument. There must be an entry for every column, as well as possible extra entries for the left and right borders of the table or for the inter-column spacings. The possible *column formatting symbols* are:

`l`	The column contents are left justified.
`c`	The column contents are centered.
`r`	The column contents are right justified.
`{wd}`	The text in this column is set into lines of width *wd* and the top line is aligned with the other columns. In fact, the text is set in a parbox with the command `\parbox[`*t*`]{`*wd*`}{`*column text*`}`.
`*{num}{cols}`	The column format contained in *cols* is reproduced *num* times, so that `*{3}{|c|}` is the same as `|c|c|c|`.

The available formatting symbols for right and left borders and for the inter-column spacing are:

`|`	Draws a vertical line.
`||`	Draws two vertical lines next to each other.
`@{text}`	This entry is referred to as an `@`-expression, and inserts text in every line of the table between the two columns where it appears.

`@`-expression removes the inter-column spacing that is automatically put between each pair of columns. If white space is needed between the inserted text and the next column, this must be explicitly included with `\hspace{ }` within the text of the `@`-expression. If the inter-column spacing between two particular columns is to be something other than the standard, this may be easily achieved by placing `@{\hspace{wd}}` between the appropriate columns in the formatting argument. This replaces the standard inter-column spacing with the width *wd*.

An `\extracolsep{`*wd*`}` within an `@`-expression will put extra spacing of amount *wd* between all the following columns, until countermanded by another `\extracolsep` command. In contrast to the standard spacing, this additional spacing is not removed by later `@`-expression. In the `\tabular*` environment, there must be a command `@{\extracolsep\fill}` somewhere in the column format so that all the subsequent inter-column spacings can stretch out to fill the predefined table width.

If the left or right borders of the table do not consist of a vertical line, a spacing equal to half the normal inter-column spacing is added there. If this spacing is not required, it may be suppressed by including an empty `@`-expression `@{}` at the beginning or end of the column format.

rows Contain the actual entries in the table, each horizontal row being terminated with \\. These rows consist of a sequence of column entries separated from each other by the & symbol. Thus each row in the table contains the same number of column entries as in the column definition *cols*. Some entries may be empty. The individual column entries are treated by LaTeXas though they were enclosed in braces { }, so that any change in type style or size are restricted to that one column.

`\hline` This command may only appear before the first row or immediately after a row termination \\. It draws a horizontal line the full width of the table below the row that was just ended, or at the top of the table if it comes at the beginning. Two `\hline` commands together draw two horizontal lines with a little space between them.

`\cline{`*n* – *m*`}`
This command draws a horizontal line from the left side of column *n* to the right side of column *m*. Like `\hline`, it may only be given just after a row termination \\, and there may be more than one after another. The command `\cline{`*1-3*`} \cline{`*5-7*`}` draws two horizontal lines from column 1 to 3 and from column 5 to 7, below the row that was just ended. In each case, the full column widths are underlined.

`\vline` This command draws a vertical line with the height of the row at the location where it appears. In this way, vertical lines that do not extend the whole height of the table may be inserted with a column.

`\multicolumn{num}{col}{text}`
This command combines the following *num* columns into a single column with their total width including inter-column spacing. The argument *col* contains exactly one of the positioning symbols `l`, `r`, `c`, with possible `@`-expressions and vertical lines `|`. A value of 1 may be given for *num* when the positioning argument is to be changed for that column in one particular row.
In this context, a 'column' starts with a positioning symbol `l`, `r`, or `c` and includes everything upto but excluding the next one. The first column also includes everything before the first positioning symbol. Thus `c@{}rl|` contains three columns: the first is `c@{}`, the second `r`, and the third `r|`.

XI.2.2. Table style parameters

There are a number of style parameters used in generating tables which LaTeXsets to standard values. These may be altered by the user, either globally within the preamble or locally inside an environment. They should not be changed within the `tabular` environment.

- `\tabcolsep` is half the width of the spacing that is inserted between columns in the `tabular` and `tabular*` environments.
- `\arrayrulewidth` is the thickness of the vertical and horizontal lines within a table.
- `\doublerulesep` is the separation between the lines of a double rule.
- `\arraystretch` can be used to change the distance between the rows of a table. This is a multiplying factor, with a standard value of `1`. A value of `1.5` means that the inter-row spacing is increased by 50%. A new value is set by redefining the parameter with the command:

```
\renewcommand{\arraystrech}{factor}
```

Following are the commands for changing the table style parameters that relate to dimensions:

```
\setlength\tabcolsep{dimen}
\setlength\arrayrulewidth{dimen}
\setlength\doublerulesep{dimen}
```

XI.2.3. Example

Creating tables is much easier in practice than it would seem from the above list of formatting possibilities. This is best illustrated with an example.

The simplest table consists of rows and columns in which the text entries are either centered or justified to one side. The column widths, the spacing between the columns, and thus the entire width of the table are automatically calculated.

Sample Tabular		
col head	col head	col head
Left	centered	right
aligned	items	aligned
items	items	items
Left items	centered	right aligned

See the code that generated the table above.

```
\begin{tabular{l|c|r|}
\hline
\multicolumn{3{|c|}{Sample Tabular}
\hline
col head   & col head & col head
\hline
Left       & centered & right   \\\cline{1-2}
aligned    & items    & aligned \\\cline{2-3}
items      & items    & items   \\\cline{1-2}
Left items & centered & right aligned
\hline
\end{tabular}
```

The discussion on tables doesn't conclude with this chapter, instead more bells and whistles are to be discussed, such as long tables (tables that span multiple pages), how to repeat the column headings and special footlines in all multipaged tables, color tables and also a few other embellishments, which the scientific community at large might require in their document preparation.

XI.2.4. Exercise

Here is an exercise you can try.

Plan for TEX Users Group 2001–2003

Project	No.		Name					
Year	2001		2002		2003			
	Rs.	US$	Rs.	US$	Rs.	US$		
Internet costs								
Journal costs								
TEXLive production costs								
Signature				Authorization				

TUTORIAL XII

CROSS REFERENCES IN LaTeX

XII.1. WHY CROSS REFERENCES?

Cross reference is the technical term for quoting yourself. This is what you do when you say something like, "As I said earlier,...". More seriously, in a written article you may often have occasion to refer the reader to something mentioned earlier (or sometimes to something yet to be said) in the same document. Thus you may have explained a new term in the second section of your article and when you use this term again in the fourth section, it is a matter of courtesy to the reader to point to the explanation. Again, in a mathematics article, you may have to cite an earlier result in the proof of the current result.

Such cross referencing can be done by hand, but if you revise your document and insert some new sections (or theorems) then changing all cross references manually is no easy task. It is always better to automate such tedious tasks. (After all what's a computer for, if not to do such mundane jobs?)

XII.2. LET LaTeX DO IT

The basic method of using cross references (see Section XII.1 for what we mean by cross reference) in LaTeX is quite simple. Suppose that somewhere in the second section of your article, you want to refer to the first section. You assign a *key* to the first section using the command

```
\section{section name}\label{key}
```

and at the point in the second section where the reference is to be made, you type the command

```
\ref{key}
```

Thus the reference "see Section XII.1..." in the first sentence of this section was produced by including the command `\label{intro}` in the command for the first section as

```
\section{Why cross references}\label{intro}
```

and the command `\ref{intro}` at the place of reference in the second section as

```
...(see Section \ref{intro} for...
```

Okay, the example is a bit silly, since the actual reference here is not *really* necessary, but you get the general idea, don't you? Incidentally, the `\label{key}` for a section need not be given immediately after the `\section{section name}`. It can be given anywhere within the section.

The first time you run LaTeX on a file named, say, `myfile.tex` containing cross references, the reference information is written in an auxiliary file named `myfile.aux` and at the end of the run LaTeX prints a warning

```
LaTeX Warning: There were undefined references.

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

A second run gets the references right. The same thing happens when you've changed the reference information in any way, say, by adding a new section.

Though the *key* in `\label{`*key*`}` can be any sequence of letters, digits or punctuation characters, it is convenient to use some mnemonic (such as `\label{`*limcon*`}` for a section entitled "Limits and Continuity" rather than `\label{`*sec@#*?!*`}`. Also, when you make a reference, it's better to type `~\ref{`*limcon*`}` (notice the *tie*?) than `\ref{`*limcon*`}` to prevent the possibility of the reference number falling off the edge as in "...see Section XII.1 for further details....".

In addition to sectioning commands such as `\chapter` or `\section`, reference can also be made to an `\item` entry in an `enumerate` environment, by attaching a `\label`. For example the input

```
In the classical \emph{syllogism}
 \begin{enumerate}
 \item All men are mortal.\label{pre1}
 \item Socrates is a man.\label{pre2}
 \item So Socrates is a mortal.\label{con}
 \end{enumerate}
Statements (\ref{pre1}) and (\ref{pre2}) are the \emph{premises} and
statement (\ref{con}) is the conclusion.
```

gives the following output

In the classical *syllogism*

(1) All men are mortal.
(2) Socrates is a man.
(3) So Socrates is a mortal.

Statements (1) and (2) are the *premises* and statement (3) is the conclusion

You must be a bit careful about references to tables or figures (technically, "floats"). For them, the `\label` command should be given after the `\caption` command or in its argument, as in the example below:

```
\begin{table}[h]
\begin{center}
\setlength{\extrarowheight}{5pt}
\begin{tabular}{|c|c|c|c|}
\hline
Value of $x$ & 1 & 2 & 3\\
\hline
Value of $y$ & 1 & 8 & 27\\
\hline
\end{tabular}
\caption{Observed values of $x$ and $y$}\label{tabxy}
```

```
\end{center}
\end{table}
Two possible relations betweeen $x$ and $y$ satisfying
the data in Table\ref{tabxy} are $y=x^3$ and
$y=6x^2-11x+6$
```

This produces the following output:

Value of x	1	2	3
Value of y	1	8	27

Table XII.1: Observed values of x and y

> Two possible relations between x and y satisfying the data in Table XII.1 are $y = x^3$ and $y = 6x^2 - 11x + 6$

You can think of a `\caption` command within a `figure` or `table` environment as a sort of sectioning command within the environment. Thus you can have several `\caption` and `\label` pairs within a single `figure` or `table` environment.

You can also make *forward* references in exactly the same way by `\ref`-ing to the *key* of some succeeding `\label` such as "see Subsection XII.2.1 for a discussion of cross references in mathematics."

XII.2.1. Cross references in math

Mathematical documents abound in cross references. There are references to theorems and equations and figures and whatnot. The method of reference is exactly as before. Thus if you've defined `\newtheorem{theorem}[subsection]`, then after typing

```
\begin{theorem}\label{diffcon}
Every differentiable function is continuous
\end{theorem}
```

you get

XII.2.1.1 Theorem. *Every differentiable function is continuous*

and you can type elsewhere in the document

```
The converse of Theorem~\ref{diffcon} is false.
```

to get

> The converse of Theorem XII.2.1.1 is false.

References can be made to equations as in the following examples:

```
\begin{equation}\label{sumsq}
  (x+y)^2=x^2+2xy+y^2
\end{equation}
Changing $y$ to $-y$ in Equation~(\ref{sumsq}) gives the following
```

(XII.1) $$(x+y)^2 = x^2 + 2xy + y^2$$

Changing y to $-y$ in Equation (XII.1) gives the following

If you load the package amsmath, you can use the command `\eqref` instead of `\ref` to make a reference to an equation. This automatically supplies the parantheses around the equation number and provides an italic correction before the closing parenthesis, if necessary. For example,

```
Equation \eqref{sumsq} gives the following .....
```

produces

Equation XII.1 gives the following .....

References can be made to individual equations in multiline displays of equations produced by such environments as `align` or `gather` (defined in the amsmath package). The `\label` command can be used within such a structure for subnumbering as in the example below:

```
\begin{align}
 (x+y)^2&=x^2+2xy+y^2\label{sum}\\
 (x-y)^2&=x^2-2xy+y^2\tag{\ref{sum}a}
\end{align}
```

(XII.2) $$(x+y)^2 = x^2 + 2xy + y^2$$

(XII.2a) $$(x-y)^2 = x^2 - 2xy + y^2$$

XII.3. POINTING TO A PAGE—THE PACKAGE VARIOREF

In making a reference to a table or an equation, it is more to convenient (for the reader, that is) to give the page number of the reference also. The command

`\pageref{`*key*`}`

typesets the number of the page where the command `\label{`*key*`}` was given. Thus for example

```
see Table~\ref{tabxy} in page~\pageref{tabxy}
```

in this document produces

see Table XII.1 in page 137

To avoid the tedium of repeated by typing

`\ref{`*key*`}` on page `\pageref{`*key*`}`

you can define the macro

```
\newcommand{\fullref}[1]{\ref{#1} on page~\pageref{#1}}
```

and use `\fullref` for such references. But the trouble is that at times the referred object and the reference to it fall on the same page (with TEX you never know this till the end) so that you get a reference to the page number of the very page you are reading, which looks funny. This can be avoided by using the package varioref. If you load this package by including `\usepackage{varioref}` in your preamble, then you can use the command

`\vref{`*key*`}`

to refer to an object you've marked with `\label{`*key*`}` elsewhere in the document. The action of `\vref` varies according to the page(s) where the referred object and the references are typeset by TEX in the final output.

(1) If the object and the reference are on the same page, `\vref` produces only a `\ref` suppressing `\pageref` so that only the number pointing to the object is typeset, without any reference to the page number.
(2) If the object and the reference are on different pages whose numbers differ by more than one, `\vref` produces both `\ref` and `\pageref`.
(3) If the object and the reference fall on pages whose numbers differ by one (that is, on successive pages), `\vref` produces `\ref` followed by the phrase "on the preceding page" or "on the following page" depending on whether the object or the reference occurs first. Moreover, in the next occurrence of `\vref` in a situation of the same type, the phrases are changed to "on the next page" and the "page before" respectively.

This is the default behavior of `\vref` in the article documentclass. If the article class is used with the twoside option or if the documentclass book is used, then the behavior in Case (3) above is a bit different.

(1) If the object and the reference fall on the two sides of the same *leaf*, the behavior of `\vref` is as in (3) above.
(2) If the object and the reference fall on pages forming a double spread (that is, a page of even number followed by the next page), then `\vref` produces `\ref` followed by the phrase "on the facing page". Moreover, in the next occurence of `\vref` in a situation of the same type, the phrases are changed to "on the preceding page" and "on the next page" respectively.

The phrases used in the various cases considered above can be customized by redefining the commands used in generating them. For the article class without the twoside option, reference to the previous page uses the command `\reftextbefore` and reference to the next page uses `\reftextafter`. In the case of the article class with the twoside option or the book class, the commands `\reftextfaceafter` and `\reftextfacebefore` are used in the case of reference to a page in a double spread. The default definitions of these commands are given below. In all these, the two arguments of the command `\reftextvario` are phrases alternatively used in the repeated use of the reference as mentioned above.

```
\newcommand{\reftextbefore}
           {on the \reftextvario{preceding page}{page before}}
\newcommand{\reftextafter}
           {on the \reftextvario{following}{next} page}
\newcommand{\reftextfacebefore}
           {on the \reftextvario{facing}{preceding} page}
\newcommand{\reftextfaceafter}
           {on the \reftextvario{facing}{next}{page}}
```

You can customize the phrases generated in various situations by redefining these with phrases of your choice in the arguments of `\reftextvario`.

If you want to refer only to a page number using `\varioref`, you can use the command

`\vpageref{`*key*`}`

to produce the page number of the object marked with `\label{`*key*`}`. The phrases used in the various special cases are the same as described above, except that when the referred object and the reference fall on the same page, either the phrase "on this page" or "on the current page" is produced. The command used to generate these is `\reftextcurrent` whose default definition is

```
\newcommand{\reftextcurrent}
           {on \reftextvario{this}{the current} page}
```

You can change the phrases "this" and "the current" *globally* by redefining this command. You can also make some *local* changes by using the two optional arguments that `\vpageref` allows. Thus you can use the command

`\vpageref[`*same page phrase*`][`*other page phrase*`]{`*key*`}`

to refer to the page number of the object marked with `\label{`*key*`}`. The *same page phrase* will be used if the object and the reference fall on the same page and the phrase *other page phrase* will be used, if they fall on different pages. Thus for example, the command

```
see the \vpageref[above table][table]{tabxy}
```

given in this document will produce

> see the above table

if the reference occurs on the same page as Table XII.1 and

> see the table on page 137

if they fall on different pages.

XII.4. POINTING OUTSIDE—THE PACKAGE XR

Sometimes you may want to refer to something in a document other than the one you are working on. (This happens, for instance if you keep an article as separate files.) The package xr allows such external references.

If you want to refer to objects in a file named `other.tex` in your current document, load the package xr and set the external document as `other.tex` using the commands

`\usepackage{xr}` `\externaldocument{`*other*`}`

in the preamble of the current document. Then you can use the `\ref` and `\pageref` to refer to anything that has been marked with the `\label` command in either the current document or `other.tex`. Any number of such external documents can be specified.

If the same *key* is used to mark different objects in two such documents, there'll be a conflict. To get over this, you can use the optional argument available in `\externaldocument` command. If you say

`\externaldocument[`*a-*`]{`*other*`}`

then a reference to `\label{`*key*`}` in `other.tex` could be made by `\ref{`*a-key*`}`. The prefix need not be `a-`; it can be any convenient string.

XII.5. LOST THE KEYS? USE lablst.tex

One of the conveniences of using keys for cross references is that you need not keep track of the actual numbers, but then you'll have to remember the keys. You can produce the list of keys used in a document by running LaTeX on the file `lablst.tex`. In our system, we do this by first typing

```
latex lablst
```

LaTeX responds as follows:

```
*********************************
* Enter input file name
*   without the .tex extension:
*********************************

\lablstfile=
```

We type in the file name as `cref` which is the source of this document and is presented with another query.

```
*********************************************
* Enter document class used in file cref.tex
*   with no options or extension:
*********************************************

\lablstclass=
```

So we type `article`. And is asked

```
*******************************************
* Enter packages used in file cref.tex
*   with no options or extensions:
*******************************************

\lablstpackages=
```

Here only those packages used in the article which define commands used in section titles etc. need be given. So we type

```
amsmath,array,enumerate
```

This produces a file `lablst.dvi` which can be viewed to see a list of keys used in the document.

Finally if your text editor is `GNU Emacs`, then you can use its `RefTeX` package to automate generation, insertion and location of keys at the editing stage.

TUTORIAL XIII

FOOTNOTES, MARGINPARS, AND ENDNOTES

LaTeX has facilities to typeset "inserted" text, such as footnotes, marginal notes, figures and tables. This chapter looks more closely at different kinds of notes.

XIII.1. FOOTNOTES

Footnotes are generated with the command

`\footnote{`*footnote_text*`}`

which comes immediately after the word requiring an explanation in a footnote. The text footnote_text appears as a footnote in a smaller typeface at the bottom of the page. The first line of the footnote is indented and is given the same footnote marker as that inserted in the main text. The first footnote on a page is separated from the rest of the page text by means of a short horizontal line.

The standard footnote marker is a small, raised number[1], which is sequentially numbered.

Footnotes produced with the `\footnote` command inside a **minipage** environment use the **mpfootnote** counter and are typeset at the bottom of the parbox produced by the **minipage**[2].

However, if you use the `\footnotemark` command in a `minipage` it will produce a footnote mark in the same style and sequence as the main text footnotes—i.e., stepping the `mpfootnote` counter and using the `\thefootnote` command for the representation. This behavior allows you to produce a footnote inside your `minipage` that is typeset in sequence with the main text footnotes at the bottom of the page: you place a `\footnotemark` inside the `minipage` and the corresponding `\footnotetext` after it. See below:

Footnotes in a minipage are numbered using lowercase letters.[a]
This text references a footnote at the bottom of the page.[3]

[a] Inside minipage

```
\begin{minipage}{5cm}
Footnotes in a minipage are numbered
using lowercase letters.\footnote{%
Inside minipage} \par This text
references a footnote at the bottom
of the page.\footnotemark
\end{minipage}
\footnotetext{At bottom of page}
```

The footnote numbering is incremented throughout the document for the article class, where it is reset to 1 for each new chapter in the report and book classes.

[1] See how the footnote is produced: " ... raised number `\footnote{See how the footnote is produced: ... }`.

[2] With nested minipages, the footnote comes after the next `\endminipage` command, which could be at the wrong place.

[3] At bottom of page.