↑ Input

```
Equation~\ref{eqn:linear} is a linear function.
\begin{equation}
\label{eqn:linear}
f(x) = mx + c
\end{equation}
```

↓ Input

↑ Output

Equation 9.2 is a linear function.

$$f(x) = mx + c \tag{9.2}$$

↓ Output

Equation numbers are usually given in parentheses, which can be done using:

```
Equation~(\ref{eqn:linear})
```

Input

The amsmath package provides a convenient short cut:

\eqref{⟨*label*⟩}

Definition

So the above can be written as:

```
Equation~\eqref{eqn:linear}
```

Input

Equation (9.2)

Output

Note:
Both the equation environment and \[...\] are only designed for one line of maths. Therefore you must not have any line breaks or paragraph breaks within them. The following will cause an error:

```
\begin{equation}

f(x) = mx + c

\end{equation}
```

✘
✘

Either remove the blank lines or comment them out:

```
\begin{equation}
%
f(x) = mx + c
%
\end{equation}
```

✔
✔

9.3 Multiple Lines of Displayed Maths

The amsmath package provides the align and align* environments for aligned equations. The starred version doesn't number the equations. These environments provide pairs of left- and right-aligned columns. As with the tabular environment, use & to separate columns and \\ to separate rows. Unlike the tabular environment, there is no argument as the column specifiers are predefined. Another difference is that no page breaks can occur in the tabular environment, but it's possible to allow a page break in align or align* using

\displaybreak[⟨*n*⟩] Definition

immediately before the \\ where it is to take effect. The optional argument is a number from 0 to 4 indicating the desirability to break the page (from 0 the least to 4 the most).

If you want to mix numbered and unnumbered rows, you can use

\notag Definition

to suppress the numbering for a particular row in the align environment. This command must go before \\ at the end of the row. The default equation numbering can be overridden for a particular row using:

\tag{⟨*tag*⟩} Definition

where ⟨*tag*⟩ is the replacement for the equation number.

⚠ Don't use the eqnarray or eqnarray* environments. They're obsolete [15].

Example (Unnumbered):

↑ Input

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

↓ Input

↑ Output

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

↓ Output

Note that the equals sign is placed at the start of the second column, *after* the ampersand &. This ensures the correct amount of spacing on either side. If the first line of the above equation was changed to:

```
y =& 2x + 2\\
```

✘

there wouldn't be enough space on the right of the equal sign:

$$y = 2x + 2$$

Example (One Row Numbered):

↑ Input
```
\begin{align}
y &= 2x + 2\notag\\
  &= 2(x+1)
\end{align}
```
↓ Input

↑ Output

$$\begin{aligned} y &= 2x + 2 \\ &= 2(x + 1) \qquad (9.3) \end{aligned}$$

↓ Output

Example (Four Columns):

↑ Input
```
\begin{align*}
y &= 2x + 2 & z &= 6x + 3\\
  &= 2(x+1) &   &= 3(2x+1)
\end{align*}
```
↓ Input

↑ Output

$$\begin{aligned} y &= 2x + 2 & z &= 6x + 3 \\ &= 2(x + 1) & &= 3(2x + 1) \end{aligned}$$

↓ Output

As with equation, you can cross-reference individual rows of an align environment, but you must remember to put `\label` *before* the end of row `\\` separator. You can reference a row in the align* environment if you have assigned it a tag with `\tag`, but don't try labelling a row in the align environment where the numbering has been suppressed with `\notag`.

Example (Cross-Referenced):
This example has two numbered equations in an align environment, both of which are labelled and referenced:

Input

```
The function $f(x)$ is given in Equation~\eqref{eq:fx}, and its
derivative $f'(x)$ is given in Equation~\eqref{eq:dfx}.
\begin{align}
f(x) &= 2x + 1 \label{eq:fx}\\
f'(x) &= 2 \label{eq:dfx}
\end{align}
```

Input

Output

The function $f(x)$ is given in Equation (9.4), and its derivative $f'(x)$ is given in Equation (9.5).

$$f(x) = 2x + 1 \tag{9.4}$$

$$f'(x) = 2 \tag{9.5}$$

Output

Recall the command `\text{⟨text⟩}` from the previous section. This can be used within cells of the align and align* environments, but the amsmath package also provides

`\intertext{⟨text⟩}` Definition

which can be used for a line of interjection between the rows. This command may only go right after `\\`.

Example

Input

```
\begin{align*}
y &= 2x + 2\\
\intertext{Using the distributive law:}
&= 2(x+1)
\end{align*}
```

Input

Output

$$y = 2x + 2$$

Using the distributive law:

$$= 2(x + 1)$$

Output

There are other environments for multiple-line displayed maths, but they are beyond the scope of this book. See the amsmath documentation for further details.

9.4 Mathematical Commands

Most of the commands described in this section may only be used in one of the mathematics environments. If you try to use a mathematics command outside a maths environment you will get a "`Missing $ inserted`" error message.

9.4.1 Maths Fonts

Just as we are able to change text fonts using the commands `\textrm`, `\textbf` etc, we can also use commands to change the maths font. Basic maths font changing commands are shown in Table 9.1.

Table 9.1 Maths Font Changing Commands

Command	**Example Input**	**Corresponding Output** (Computer Modern)
`\mathrm{`⟨*maths*⟩`}`	`$\mathrm{xyz}$`	xyz
`\mathsf{`⟨*maths*⟩`}`	`$\mathsf{xyz}$`	xyz
`\mathtt{`⟨*maths*⟩`}`	`$\mathtt{xyz}$`	$\mathtt{xyz}$
`\mathit{`⟨*maths*⟩`}`	`$\mathit{xyz}$`	xyz
`\mathbf{`⟨*maths*⟩`}`	`$\mathbf{xyz}$`	$\mathbf{xyz}$
`\mathcal{`⟨*maths*⟩`}`	`$\mathcal{XYZ}$`	$\mathcal{XYZ}$

The calligraphic fonts via `\mathcal` are only available for upper-case characters. Table 9.2 lists additional font commands supplied with the amsmath and amsfonts packages.

[FAQ: Better script fonts for maths]

Table 9.2 The amsfonts‡ and amsmath† Font Commands

Command	**Example Input**	**Example Output**
‡`\mathbb{`⟨*maths*⟩`}`	`$\mathbb{A+B=C}$`	$\mathbb{A}+\mathbb{B}=\mathbb{C}$
‡`\mathfrak{`⟨*maths*⟩`}`	`$\mathfrak{A+B=C}$`	$\mathfrak{A}+\mathfrak{B}=\mathfrak{C}$
†`\boldsymbol{`⟨*maths*⟩`}`	`$\boldsymbol{A+B=C}$`	$\boldsymbol{A+B=C}$
†`\pmb{`⟨*symbol*⟩`}`	`$\pmb{+-=}$`	$\pmb{+-=}$

9.4.2 Greek Letters

Greek letters that differ from the corresponding Roman letters are obtained by placing a backslash in front of the name.[9.1] Lower case and upper case Greek letters are shown in Table 9.3 and Table 9.4, respectively. There are also some variants of certain symbols, such as `\vartheta` as opposed to `\theta`.

[9.1] So, for example, there is no omicron since it looks the same as a Roman o.

Table 9.3 Lower Case Greek Letters

\alpha	α	\beta	β	\gamma	γ
\delta	δ	\epsilon	ϵ	\varepsilon	ε
\zeta	ζ	\eta	η	\theta	θ
\vartheta	ϑ	\iota	ι	\kappa	κ
\lambda	λ	\mu	μ	\nu	ν
\xi	ξ	\pi	π	\varpi	ϖ
\rho	ρ	\varrho	ϱ	\sigma	σ
\varsigma	ς	\tau	τ	\upsilon	υ
\phi	ϕ	\varphi	φ	\chi	χ
\psi	ψ	\omega	ω		

Table 9.4 Upper Case Greek Letters

\Gamma	Γ	\Delta	Δ	\Theta	Θ
\Lambda	Λ	\Xi	Ξ	\Pi	Π
\Sigma	Σ	\Upsilon	Υ	\Phi	Φ
\Psi	Ψ	\Omega	Ω		

Example:
The following code

```
\[ x' = x + \Delta x \]
```
Input

produces:

$$x' = x + \Delta x$$
Output

9.4.3 Subscripts and Superscripts

Subscripts are obtained either by the command

\sb{⟨*maths*⟩} Definition

or by the special character:

_{⟨*maths*⟩} Definition

Superscripts are obtained either by the command

\sp{⟨*maths*⟩} Definition

or by the special character:

^{⟨*maths*⟩} Definition

Examples:

1. This example uses \sb and \sp:

```
\[y = x\sb{1}\sp{2} + x\sb{2}\sp{2}\]
```
Input

2. This example uses `_` and `^`

```
\[y = x_{1}^{2} + x_{2}^{2}\]
```
Input

3. Recall from page 16 that mandatory arguments only consisting of one character don't need to be grouped, so the above code can also be written as:

```
\[y = x_1^2 + x_2^2\]
```
Input

   This is simpler than the first two examples. However it's a good idea to be in the habit of always using braces in case you forgot them when they're needed.

All three of the above examples produce the same output:

$$y = x_1^2 + x_2^2$$
Output

Notice how the subscript gets tucked under the slope of the Y in:

```
\[ Y_{1}^{2} \]
```
Input

$$Y_{1}^{2}$$
Output

Compare with

```
\[ Y{}_{1}^{2} \]
```
Input

$$Y{}_{1}^{2}$$
Output

Example (Nested)
Subscripts and superscripts can also be nested (note that it is now necessary to group the argument to the superscript command):

```
\[ f(x) = e^{x_1} \]
```
Input

which produces

$$f(x) = e^{x_1}$$
Output

This example isn't quite right as e isn't actually a variable and shouldn't be typeset in italic. The correct way to do this is:

```
\[ f(x) = \mathrm{e}^{x_1} \]
```
Input

which results in:

$$f(x) = \mathrm{e}^{x_1}$$
Output

If you are going to use e a lot, it will be simpler to define a new command to do this. The definition should go in the preamble:

```
\newcommand{\e}{\mathrm{e}}
```
Input

Then in the document:

```
\[ f(x_1, x_2) = \e^{x_1^2} + \e^{x_2^2} \]
```
Input

$$f(x_1, x_2) = \mathrm{e}^{x_1^2} + \mathrm{e}^{x_2^2}$$
Output

Take care when nesting subscripts or superscripts. The following

```
x_1_2
```
✗

will give a `! Double subscript` error.

9.4.4 Functional Names

Functions such as log and tan can't simply be typed in as `log` or `tan` otherwise they will come out looking like the variables l times o times g (log) or t times a times n (tan). Instead you should use one of the commands listed in Table 9.5. The functions denoted with † can have limits by using the subscript command `_` or the superscript command `^`. In addition, the modulo commands listed in Table 9.6 are also available.

[FAQ: Sub- and superscript positioning for operators]

Table 9.5 Function Names (†indicates command may have limits, ‡defined by amsmath).

`\arccos`	$\arccos$	`\arcsin`	$\arcsin$	`\arctan`	$\arctan$
`\arg`	$\arg$	`\cos`	$\cos$	`\cosh`	$\cosh$
`\cot`	$\cot$	`\coth`	$\coth$	`\csc`	$\csc$
`\deg`	$\deg$	`\det`†	$\det$	`\dim`	$\dim$
`\exp`	$\exp$	`\gcd`†	$\gcd$	`\hom`	$\hom$
`\inf`†	$\inf$	`\injlim`†‡	$\injlim$	`\ker`	$\ker$
`\lg`	$\lg$	`\lim`†	$\lim$	`\liminf`†	$\liminf$
`\limsup`†	$\limsup$	`\ln`	$\ln$	`\log`	$\log$
`\max`†	$\max$	`\min`†	$\min$	`\Pr`†	$\Pr$
`\projlim`†‡	$\projlim$	`\sec`	$\sec$	`\sin`	$\sin$
`\sinh`	$\sinh$	`\sup`†	$\sup$	`\tan`	$\tan$
`\tanh`	$\tanh$	`\varinjlim`†‡	$\varinjlim$	`\varliminf`†‡	$\varliminf$
`\varlimsup`†‡	$\varlimsup$	`\varprojlim`†‡	$\varprojlim$		

Table 9.6 Modulo Commands (‡defined by amsmath package)

Command	**Example Input**	**Example Output**
`\bmod`	`$m \bmod n$`	$m \bmod n$
`\pmod{`⟨*maths*⟩`}`	`$m \pmod{n}$`	$m \pmod{n}$
`\mod{`⟨*maths*⟩`}`‡	`$m \mod{n}$`	$m \mod{n}$
`\pod{`⟨*maths*⟩`}`‡	`$m \pod{n}$`	$m \pod{n}$

Example (Trigonometric Functions):
This example uses the cos and sin functions and also the Greek letter theta.

```
\[ z = r(\cos\theta + i\sin\theta) \]
```
Input

$$z = r(\cos\theta + i\sin\theta)$$
Output

Example (Limit):
The command `\infty` is the infinity symbol ∞, and the command `\to` displays an arrow pointing to the right. Note the use of `_` since the limit is a subscript.

```
\[ \lim_{x\to\infty} f(x) \]
```
Input

$$\lim_{x\to\infty} f(x)$$
Output

The operators with limits behave differently depending on whether they are in displayed or in-line maths. Notice the difference when the same code appears in-line:

```
In a line of text $\lim_{x\to\infty} f(x)$
```
Input

which now displays as:

In a line of text $\lim_{x\to\infty} f(x)$ Output

Example (With Subscript):
This is another example of a functional name using a subscript:

```
\[ \min_x f(x) \]
```
Input

$$\min_x f(x)$$
Output

Again, notice the difference when it is used in-line:

```
In a line of text $\min_x f(x)$
```
Input

In a line of text $\min_x f(x)$ Output

Defining New Functional Operators

It may be that you want a function that isn't specified in Table 9.5. In this case, the amsmath provides the ***preamble only*** command

\DeclareMathOperator{⟨*cmd*⟩}{⟨*operator name*⟩} Definition

or its starred variant

\DeclareMathOperator*{⟨*cmd*⟩}{⟨*operator name*⟩} Definition

Both versions define a command called ⟨*cmd*⟩, which must start with a backslash, that typesets ⟨*operator name*⟩ as a function name. The starred version is for function names that can take limits (like `\lim` and `\min` described above).

[FAQ: Defining a new log-like function in LaTeX]

Example (Operator Without Limits):
Suppose I want a function called card, which represents the cardinality of a set $\mathcal{S}$. First I need to define the new operator command (which I'm going to call \card) *in the preamble*:

```
\DeclareMathOperator{\card}{card}
```
Input

This operator doesn't take any limits, so I have used the unstarred version.

Later in the document, I can use this new operator command:

```
\[ n = \card(\mathcal{S}) \]
```
Input

$$n = \operatorname{card}(\mathcal{S})$$
Output

In this example `\mathcal` is used as sets are typically represented in a calligraphic font.

Example (Operator With Limits):
Suppose I now want a function called mode, which represents the mode of a set of numbers. First, I define the operator command in the preamble:

```
\DeclareMathOperator*{\mode}{mode}
```
Input

This operator needs to be able to have a subscript, so I have used the starred version.

Later in the document, I can use this new operator command:

```
\[ x_m = \mode_{x \in \mathcal{S}}(x) \]
```
Input

$$x_m = \operatorname*{mode}_{x \in \mathcal{S}}(x)$$
Output

9.4.5 Fractions

Fractions are created using the command

`\frac{⟨numerator⟩}{⟨denominator⟩}` Definition

The amsmath package also provides the command

`\cfrac[⟨pos⟩]{⟨numerator⟩}{⟨denominator⟩}` Definition

which is designed for continued fractions. The optional argument `pos` can be used for left (`l`) or right (`r`) placement of any of the numerators. (The default is centred.)

Example:
A simple fraction:

```
\[ \frac{1}{1+x} \]
```
Input

Produces:

$$\frac{1}{1+x}$$
Output

Compare with:

```
In-line: $ \frac{1}{1+x} $
```
Input

which produces:

In-line: $\frac{1}{1+x}$
Output

Example (Nested):

```
\[ \frac{1+\frac{1}{x}}{1+x+x^2} \]
```
Input

$$\frac{1+\frac{1}{x}}{1+x+x^2}$$
Output

Example (Continued Fraction):
A continued fraction (example taken from amsmath documentation and uses `\sqrt`, described in Section 9.4.6, and `\dotsb`, described in Section 9.4.7):

↑ Input

```
\[
\cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+\dotsb
}}}
\]
```

↓ Input

↑ Output

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\dotsb}}}$$

↓ Output

Example (A Derivative):

```
\[ f'(x) = \frac{df}{dx} \]
```

Input

$$f'(x) = \frac{df}{dx}$$

Output

As with "e", the differential operator "d" should be in an upright font as it is not a variable:

↑ Input

```
\[
  f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}
\]
```

↓ Input

$$f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}$$

Output

The above example is rather cumbersome, particularly if you have a lot of derivatives, so it might be easier to define a new command (see Chapter 8 (Defining Commands)). In the preamble define:

```
\newcommand{\deriv}[2]{\frac{\mathrm{d}#1}{\mathrm{d}#2}}
```

Input

Then in the document:

```
\[ f'(x) = \deriv{f}{x} \]
```

Input

$$f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}$$

Output

Example (Partial Derivative):
Partial derivatives can be obtained similarly using the command `\partial` to display the partial derivative symbol. As in the previous example, first define a new command to format a partial derivative in the preamble:

```
\newcommand{\pderiv}[2]{\frac{\partial #1}\partial #2}
```
Input

Then in the document:

```
\[ f_x = \pderiv{f}{x} \]
```
Input

$$f_x = \frac{\partial f}{\partial x}$$

Output

Example (Double Partial Derivative):

Input

```
\[
  f_{xy} = \frac{\partial^2 f}{\partial x \partial y}
\]
```

Input

$$f_{xy} = \frac{\partial^2 f}{\partial x \partial y}$$

Output

Example (First principles):

Input

```
\[
    f'(x) = \lim_{\Delta x \to 0}
    \frac{f(x + \Delta x)-f(x)}{\Delta x}
\]
```

Input

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Output

9.4.6 Roots

Roots are obtained using the command

\sqrt[⟨*order*⟩]{⟨*maths*⟩} Definition

without the optional argument ⟨*order*⟩ it will produce a simple square root. Cubic roots etc can be obtained using the optional argument.

Examples:

1. A square root:

   ```
   \[ \sqrt{a+b} \]
   ```
   Input

   $$\sqrt{a+b}$$

   Output

2. A cubic root:

   ```
   \[ \sqrt[3]{a+b} \]
   ```
   Input

   $$\sqrt[3]{a+b}$$

   Output

3. An nth root:

```
\[ \sqrt[n]{a+b} \]
```
Input

$$\sqrt[n]{a+b}$$
Output

9.4.7 Mathematical Symbols

Relational symbols are shown in Table 9.7. If you want a negation that is not shown, you can obtain it by preceding the symbol with the command `\not`. For example: `\not\subset` produces the symbol $\not\subset$.

[FAQ: Where can I find the symbol for …]

Table 9.7 Relational Symbols

`\approx`	$\approx$	`\asymp`	$\asymp$	`\bowtie`	$\bowtie$
`\cong`	$\cong$	`\dashv`	$\dashv$	`\doteq`	$\doteq$
`\equiv`	$\equiv$	`\frown`	$\frown$	`\ge` or `\geq`	$\geq$
`\gg`	$\gg$	`\in`	$\in$	`\le` or `\leq`	$\leq$
`\ll`	$\ll$	`\mid` or `|`	$\mid$	`\models`	$\models$
`\neq`	$\neq$	`\ni`	$\ni$	`\notin`	$\notin$
`\parallel`	$\parallel$	`\prec`	$\prec$	`\preceq`	$\preceq$
`\perp`	$\perp$	`\propto`	$\propto$	`\sim`	$\sim$
`\simeq`	$\simeq$	`\smile`	$\smile$	`\sqsubseteq`	$\sqsubseteq$
`\sqsupseteq`	$\sqsupseteq$	`\subset`	$\subset$	`\subseteq`	$\subseteq$
`\succ`	$\succ$	`\succeq`	$\succeq$	`\supset`	$\supset$
`\supseteq`	$\supseteq$	`\vdash`	$\vdash$		

Binary operator symbols are shown in Table 9.8, and arrow symbols are shown in Table 9.9. There are also over and under arrows (Table 9.10) that have an argument. The over arrows put an extendible arrow over their argument, and the under arrows put an extendible arrow under their argument. In addition, the amsmath package provides extensible arrows that take a superscript and, optionally, a subscript:

`\xleftarrow[`⟨*subscript*⟩`]{`⟨*superscript*⟩`}` Definition

`\xrightarrow[`⟨*subscript*⟩`]{`⟨*superscript*⟩`}` Definition

Example:

Input
```
\[
   A \xleftarrow{n+m-p} B \xrightarrow[X]{n+p} C
\]
```
Input

Output

$$A \xleftarrow{n+m-p} B \xrightarrow[X]{n+p} C$$

↓ Output

Table 9.8 Binary Operator Symbols

\amalg	$\amalg$	\ast	$\ast$	\bullet	$\bullet$
\bigcirc	$\bigcirc$	\bigtriangledown	$\bigtriangledown$	\bigtriangleup	$\bigtriangleup$
\cap	$\cap$	\cdot	$\cdot$	\circ	$\circ$
\cup	$\cup$	\dagger	$\dagger$	\ddagger	$\ddagger$
\diamond	$\diamond$	\div	$\div$	\mp	$\mp$
\odot	$\odot$	\ominus	$\ominus$	\oplus	$\oplus$
\oslash	$\oslash$	\otimes	$\otimes$	\pm	$\pm$
\setminus	$\setminus$	\sqcap	$\sqcap$	\sqcup	$\sqcup$
\star	$\star$	\times	$\times$	\triangleleft	$\triangleleft$
\triangleright	$\triangleright$	\uplus	$\uplus$	\vee	$\vee$
\wedge	$\wedge$	\wr	$\wr$		

Table 9.9 Arrow Symbols

\downarrow	$\downarrow$	\Downarrow	$\Downarrow$
\hookleftarrow	$\hookleftarrow$	\hookrightarrow	$\hookrightarrow$
\leftarrow or \gets	$\leftarrow$	\Leftarrow	$\Leftarrow$
\leftharpoondown	$\leftharpoondown$	\leftharpoonup	$\leftharpoonup$
\leftrightarrow	$\leftrightarrow$	\Leftrightarrow	$\Leftrightarrow$
\longleftarrow	$\longleftarrow$	\Longleftarrow	$\Longleftarrow$
\longleftrightarrow	$\longleftrightarrow$	\Longleftrightarrow	$\Longleftrightarrow$
\longmapsto	$\longmapsto$	\longrightarrow	$\longrightarrow$
\Longrightarrow	$\Longrightarrow$	\mapsto	$\mapsto$
\nearrow	$\nearrow$	\nwarrow	$\nwarrow$
\rightarrow or \to	$\rightarrow$	\Rightarrow	$\Rightarrow$
\rightharpoondown	$\rightharpoondown$	\rightharpoonup	$\rightharpoonup$
\rightleftharpoons	$\rightleftharpoons$	\searrow	$\searrow$
\swarrow	$\swarrow$	\uparrow	$\uparrow$
\Uparrow	$\Uparrow$	\updownarrow	$\updownarrow$
\Updownarrow	$\Updownarrow$		

Symbols that can have limits are shown in Table 9.11. The size of these symbols depends on whether they are in displayed maths or in-line maths.

Example (Displayed Summation and Product):
The limits of summations and products are placed above and below the symbol in displayed maths:

↑ Input

```
\[
  f(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i
\]
```

Table 9.10 Over and Under Arrows (†defined by amsmath)

Definition	Example	
\overleftarrow{⟨maths⟩}	\overleftarrow{ABC}	$\overleftarrow{ABC}$
\overrightarrow{⟨maths⟩}	\overrightarrow{ABC}	$\overrightarrow{ABC}$
\overleftrightarrow{⟨maths⟩}†	\overleftrightarrow{ABC}	$\overleftrightarrow{ABC}$
\underleftarrow{⟨maths⟩}†	\underleftarrow{ABC}	$\underleftarrow{ABC}$
\underrightarrow{⟨maths⟩}†	\underrightarrow{ABC}	$\underrightarrow{ABC}$
\underleftrightarrow{⟨maths⟩}†	\underleftrightarrow{ABC}	$\underleftrightarrow{ABC}$

Table 9.11 Symbols with Limits

\sum	$\sum$	\int	$\int$	\oint	$\oint$
\prod	$\prod$	\coprod	$\coprod$	\bigcap	$\bigcap$
\bigcup	$\bigcup$	\bigsqcup	$\bigsqcup$	\bigvee	$\bigvee$
\bigwedge	$\bigwedge$	\bigodot	$\bigodot$	\bigotimes	$\bigotimes$
\bigoplus	$\bigoplus$	\biguplus	$\biguplus$		

↓ Input

$$f(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i$$

Output

Example (In-line Summation and Product):
The limits of summations and products are placed to the right of the symbol in in-line maths:

↑ Input

```
In a line of text:
\begin{math}
  f(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i
\end{math}
```

↓ Input

In a line of text: $f(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i$

Output

Multiline Sub- or Superscripts
The amsmath package provides the command:

\substack{⟨maths⟩}

Definition

which can be used for multiline sub- or superscripts. Within the argument ⟨maths⟩ use \\ to separate rows. For example:

↑ Input

```
\[
 \sum_{
```

```
  \substack
  {
    i \in \mathcal{I}\\
    i \neq 0
  }
 }
 x_i
\]
```

↓ Input

↑ Output

$$\sum_{\substack{i\in\mathcal{I}\\ i\neq 0}} x_i$$

↓ Output

9.4.8 Ellipses

Ellipsis (omission mark) commands are shown in Table 9.12. The amsmath package also provides: `\dotsc` for dots with commas, `\dotsb` for dots with binary operators/relations, `\dotsm` for multiplication dots, `\dotsi` for dots with integrals and `\dotso` for other dots, which can be used as replacements for `\ldots` and `\cdots`.

Table 9.12 Ellipses († provided by amsmath package)

`\vdots`	$\vdots$	`\cdots`	$\cdots$	`\dotsb`†	$\cdots$	`\dotsi`†	$\cdots$	`\dotsm`†	$\cdots$
`\ddots`	$\ddots$	`\ldots`	$\ldots$	`\dotsc`†	$\ldots$	`\dotso`†	$\ldots$		

Example (Low Ellipsis):
This example uses the command `\forall` to produce the "for all" symbol ∀, and it also uses `\␣` (backslash space) to make a space before the for all symbol. The amsmath "dots with commas" ellipsis `\dotsc` is used rather than the standard `\ldots`:

↑ Input

```
\[
a_ix_i = b_i\␣\forall i = 1,\dotsc, n
\]
```

↓ Input

$$a_ix_i = b_i\ \forall i = 1,\dotsc, n$$

Output

Example (Centred ellipsis):
This example uses the amsmath "dots with binary operators/relations" `\dotsb` instead of the standard `\cdots`:

↑ Input

```
\[
y = a_1 + a_2 + \dotsb + a_n
\]
```

↓ Input

$$y = a_1 + a_2 + \dots + a_n$$

Output

Exercise 22 (Maths: Fractions and Symbols)

This exercise uses a fraction, a square root, subscripts, superscripts and symbols. Try to reproduce the following output:

↑ Output

The quadratic equation

$$\sum_{i=0}^{2} a_i x^i = 0$$

has solutions given by

$$x = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2}$$

↓ Output

Again you can download or view the solution.

9.4.9 Delimiters

Placing brackets around a tall object in maths mode, such as fractions, does not look right if you use normal sized brackets. For example:

↑ Input

```
\[
  (\frac{1}{1+x})
\]
```

↓ Input

results in:

$$(\frac{1}{1+x})$$

Output

Instead, you can automatically resize the delimiters using the commands:

`\left`⟨*delimiter*⟩ Definition

and

`\right`⟨*delimiter*⟩ Definition

Rewriting the above example:

↑ Input

```
\[
  \left( \frac{1}{1+x} \right)
\]
```

↓ Input

produces:

$$\left(\frac{1}{1+x}\right)$$

Output

Note that you must always have matching `\left` and `\right` commands, although the delimiters used may be different. If you want one of the delimiters to be invisible, use a `.` (full stop) as the delimiter. Available delimiters are shown in Table 9.13. (Note for a vertical bar delimiter it's best to use amsmath's `\lvert` command instead of `|` and `\lVert` instead of `\|`.) Sometimes using `\left` and `\right` doesn't produce the optimal sized delimiters. In which case you can use additional commands provided by the amsmath package shown in Table 9.14.

Table 9.13 Delimiters (†defined by amsmath)

`(`	$($	`)`	$)$	`[`	$[$	`]`	$]$
`\{`	$\{$	`\}`	$\}$	`\lvert`†	$\lvert$	`\rvert`†	$\rvert$
`\lVert`†	$\lVert$	`\rVert`†	$\rVert$	`\langle`	$\langle$	`\rangle`	$\rangle$
`\lfloor`	$\lfloor$	`\rfloor`	$\rfloor$	`\lceil`	$\lceil$	`\rceil`	$\rceil$
`\uparrow`	$\uparrow$	`\downarrow`	$\downarrow$	`\Uparrow`	$\Uparrow$	`\Downarrow`	$\Downarrow$
`\updownarrow`	$\updownarrow$	`\Updownarrow`	$\Updownarrow$	`/`	$/$	`\backslash`	$\backslash$

Example (Vertical Bar Delimiters):

↑ Input

```
\[
\left\lvert
\frac{1}{1+x}
\right\rvert
\]
```

↓ Input

$$\left\lvert\frac{1}{1+x}\right\rvert$$

Output

Example (Delimiter with Subscript):
Delimiters can take limits:

↑ Input

```
\[
\left\lvert
\frac{1}{1+x}
\right\rvert_{x=0}
\]
```

↓ Input

Table 9.14 Additional Commands Provided by amsmath for Delimiter Sizing

Definitions		Example	
Default Size		`$( X )$`	(X)
`\bigl`⟨*delim*⟩	`\bigr`⟨*delim*⟩	`$\bigl( X \bigr)$`	$\bigl(X\bigr)$
`\Bigl`⟨*delim*⟩	`\Bigr`⟨*delim*⟩	`$\Bigl( X \Bigr)$`	$\Bigl(X\Bigr)$
`\biggl`⟨*delim*⟩	`\biggr`⟨*delim*⟩	`$\bigl( X \biggr)$`	$\biggl(X\biggr)$
`\Biggl`⟨*delim*⟩	`\Biggr`⟨*delim*⟩	`$\Biggl( X \Biggr)$`	$\Biggl(X\Biggr)$

$$\left|\frac{1}{1+x}\right|_{x=0}$$

Output

Example (Mismatch):
The left and right delimiters don't have to match:

↑ Input

```
\[
\left[\frac{1}{1+x}\right\rangle
\]
```

↓ Input

$$\left[\frac{1}{1+x}\right\rangle$$

Output

Example (An invisible delimiter):
Every `\right` must have a matching `\left` (and vice versa), so use a `.` (full stop) for an invisible delimiter.

↑ Input

```
\[
 \left.
  \frac{\partial f}{\partial x}
 \right\rvert_{x=0}
\]
```

↓ Input

$$\left.\frac{\partial f}{\partial x}\right\rvert_{x=0}$$

Output

We have now covered enough to reproduce the equation shown in Chapter 1 (Introduction):

Input

```
\newcommand*{\pderiv}[2]{\frac{\partial #1}{\partial #2}}
\newcommand*{\e}{\mathrm{e}}

\[
\pderiv{^2\mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
  \pderiv{v_i}{\rho_i} \frac{\e^{v_i}}{1-\e^{v_i}}
  + v_i \frac{\e^{v_i}\pderiv{v_i}{\rho_i}(1-\e^{v_i})
           +\e^{2v_i}\pderiv{v_i}{\rho_i}}{(1-\e^{v_i})^2}
\right)
\]
```

Input

Output

$$\frac{\partial^2\mathcal{L}}{\partial {z_i^\rho}^2} = -\frac{\partial \rho_i}{\partial z_i^\rho}\left(\frac{\partial v_i}{\partial \rho_i}\frac{\mathrm{e}^{v_i}}{1-\mathrm{e}^{v_i}} + v_i\frac{\mathrm{e}^{v_i}\frac{\partial v_i}{\partial \rho_i}(1-\mathrm{e}^{v_i})+\mathrm{e}^{2v_i}\frac{\partial v_i}{\partial \rho_i}}{(1-\mathrm{e}^{v_i})^2}\right)$$

Output

Note:
The above code looks a bit complicated, and there are so many braces that it can be easy to lose track, so here are some ways of making it a little easier to type:

1. Whenever you start a new environment type in the `\begin` and `\end` bits first, and then insert whatever goes inside the environment. This ensures that you always have a matching `\begin` and `\end`. The same goes for `\[` and `\]`.
2. Whenever you type any braces, always type the opening and closing braces first, and then insert whatever goes in between. This will ensure that your braces always match up.

So keeping these notes in mind, let's try typing in the code in a methodical manner:

1. Start and end the displayed maths mode:

   Input

   ```
   \[
   \]
   ```

   Input

2. We now need a partial derivative. (The command `\pderiv` is defined as described earlier on page 153. Make sure you remember to define it, preferably in the preamble.)

```
\[
\pderiv{}{}
\]
```

3. Let's do the first argument. This partial derivative is actually a double derivative, which means we need a squared bit on the top along with a calligraphic L:

```
\[
\pderiv{^2 \mathcal{L}}{}
\]
```

4. The second argument is the z_i^ρ squared bit. This is a nested superscript `{z_i^\rho}^2`:

```
\[
\pderiv{^2 \mathcal{L}}{{z_i^\rho}^2}
\]
```

5. We can do the next partial derivative in the same way. This one is slightly easier to do:

```
\[
\pderiv{^2 \mathcalL}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\]
```

6. Delimiters also need to occur in pairs, like curly braces and `\begin` and `\end`, so let's do them next:

```
\[
\pderiv{^2 \mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
  \left(
  \right)
\]
```

7. Now we need to do the bits inside the brackets. First of all we have yet another partial derivative:

↑ Input

```
\[
\pderiv{^2 \mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
  \left(
    \pderiv{v_i}{\rho_i}
  \right)
\]
```

↓ Input

8. Now we have a fraction following the partial derivative from the previous step. (Make sure you use braces for the exponential bit: `\e^{v_i}` (e^{v_i}) is not the same as `\e^v_i` (e^v_i). The command `\e` is defined as described earlier in Section 9.4.3. Make sure you define it, preferably in the preamble.)

↑ Input

```
\[
\pderiv{^2 \mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
  \left(
    \pderiv{v_i}{\rho_i} \frac{\e^{v_i}}{1-\e^{v_i}}
  \right)
\]
```

↓ Input

9. This is followed by v_i times another fraction:

↑ Input

```
\[
\pderiv{^2 \mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
  \left(
    \pderiv{v_i}{\rho_i} \frac{\e^{v_i}}{1-\e^{v_i}}
    + v_i \frac{}{}
  \right)
\]
```

↓ Input

10. The bottom part of the fraction (the denominator) is easier than the top, so let's do that first:

↑ Input

```
\[
\pderiv{^2 \mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
  \left(
    \pderiv{v_i}{\rho_i} \frac{\e^{v_i}}{1-\e^{v_i}}
    + v_i \frac{}{(1-\e^{v_i})^2}
  \right)
\]
```

↓ Input

11. Now for the top part of the fraction (the numerator). To refresh your memory, it should look like:

$$e^{v_i}\frac{\partial v_i}{\partial \rho_i}(1 - e^{v_i}) + e^{2v_i}\frac{\partial v_i}{\partial \rho_i}$$

That's a bit complicated, so let's break it down:

a) The first term is:

`\e^{v_i}`

b) The next term is another partial derivative:

`\pderiv{v_i}{\rho_i}`

c) Then we have:

`(1-\e^{v_i})`

d) Next we have to add on:

`+\e^{2v_i}`

e) And finally we have:

`\pderiv{v_i}{\rho_i}`

So the numerator is:

↑ Input

```
\e^{v_i}\pderiv{v_i}{\rho_i}(1-\e^{v_i})
  + \e^{2v_i}\pderiv{v_i}{\rho_i}
```

↓ Input

Inserting this into our code:

↑ Input

```
\[
\pderiv{^2\mathcal{L}}{{z_i^\rho}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
  \pderiv{v_i}{\rho_i} \frac{\e^{v_i}}{1-\e^{v_i}}
  + v_i \frac{\e^{v_i}\pderiv{v_i}{\rho_i}(1-\e^{v_i})
             +\e^{2v_i}\pderiv{v_i}{\rho_i}}{(1-\e^{v_i})^2}
\right)
\]
```

↓ Input

9.4.10 Arrays

Mathematical structures such as matrices and vectors require elements to be arranged in rows and columns. Just as we can align material in rows and columns in text mode using the tabular environment (Section 4.6), we can do the same in maths mode using the array environment. The array environment has the same format as the tabular environment, however it must be in maths mode. The column half-gaps are given by the length register `\arraycolsep` (analogous to `\tabcolsep`).

Example:

↑ Input

```
\[
\begin{array}{rrr}
0 & 1 & 19\\
-6 & 10 & 200
\end{array}
\]
```

↓ Input

↑ Output

$$\begin{array}{rrr}
0 & 1 & 19\\
-6 & 10 & 200
\end{array}$$

↓ Output

Example (Adding Delimiters):

↑ Input

```
\[
\left(
 \begin{array}{rrr}
   0 & 1 & 19\\
  -6 & 10 & 200
```

```
 \end{array}
\right)
\]
```

Input

$$\left(\begin{array}{rrr} 0 & 1 & 19\\ -6 & 10 & 200 \end{array}\right)$$

Output

Adding a Vertical Rule:
A vertical rule can be added using | in the column specifier. For example:

Input

```
\[
\left(
 \begin{array}{rr|r}
   0 & 1 & 19\\
  -6 & 10 & 200
 \end{array}
\right)
\]
```

Input

$$\left(\begin{array}{rr|r} 0 & 1 & 19\\ -6 & 10 & 200 \end{array}\right)$$

Output

Example (Cases):
This example uses an invisible delimiter:

Input

```
\[
f(x) =
 \left\{
  \begin{array}{rl}
   -1 & x < 0\\
   0  & x = 0\\
   +1 & x > 0
  \end{array}
 \right.
\]
```

Input

$$f(x) = \left\{ \begin{array}{rl} -1 & x < 0\\ 0 & x = 0\\ +1 & x > 0 \end{array} \right.$$

Output

This can be rewritten more compactly using the amsmath cases environment:

Input

```
\[
f(x) =
 \begin{cases}
  -1 & x < 0\\
```

```
  0 & x = 0\\
  +1 & x > 0
 \end{cases}
\]
```

↓ Input

$$f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ +1 & x > 0 \end{cases}$$

Output

The amsmath package provides some convenient environments to typeset matrices: pmatrix, bmatrix, Bmatrix, vmatrix and Vmatrix. These are similar to the array environment except there is no argument, and they add (respectively) ( ), [ ], { }, | | and ‖ ‖ delimiters. There is also the matrix environment that doesn't have any delimiters.

Example:

↑ Input

```
\begin{equation}
 \begin{pmatrix}
  a & b\\
  c & d
 \end{pmatrix}
\end{equation}
```

↓ Input

↑ Output

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{9.6}$$

↓ Output

The amsmath package also provides the environment smallmatrix designed for in-line use. You need to add any delimiters explicitly.

Example:

↑ Input

```
Here is a small matrix
\begin{math}
 \left(
  \begin{smallmatrix}
  a & b\\
  c & d
  \end{smallmatrix}
 \right)
\end{math}
in a line of text.
```

↓ Input

Here is a small matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ in a line of text.

Output

9.4.11 Vectors

A variable representing a vector can be typeset using the command:

`\vec{⟨variable⟩}` Definition

Example:

```
\[ \vec{x} \]
```
Input

$$\vec{x}$$
Output

Vectors are often typeset in bold. This can be done by redefining the `\vec` command. You could use `\mathbf`, for example:

```
\renewcommand{\vec}[1]{\mathbf{#1}}

\[
  \vec{x}\cdot\vec{\xi} = z
\]
```
Input

$$\mathbf{x}\cdot\xi = z$$
Output

However, as you may have noticed, the Greek letter ξ has not come out in bold. Here's an alternative (using `\boldsymbol` defined in the amsfonts package):

```
\renewcommand{\vec}[1]{\boldsymbol{#1}}
\[
  \vec{x}\cdot\vec{\xi} = z
\]
```
Input

$$\boldsymbol{x}\cdot\boldsymbol{\xi} = z$$
Output

Located (or position) vectors, on the other hand, are usually typeset with a right arrow, but the default definition of `\vec` produces an arrow that is too small:

```
\[ \vec{OP} \]
```
Input

$$\vec{OP}$$
Output

Instead, use `\overrightarrow` (Table 9.10):

```
\[ \overrightarrow{OP} \]
```
Input

$$\overrightarrow{OP}$$
Output

You might prefer to define separate commands for a located vector and a vector variable.

Example:
In the preamble, define \lvec for a located vector and \bvec for a vector variable:

↑ Input
```
\newcommand*{\lvec}[1]{\overrightarrow{#1}}
\newcommand*{\bvec}[1]{\boldsymbol{#1}}
```
↓ Input

Later in the document:

↑ Input
```
Let $\bvec{u}=(x, y)$ represent $\lvec{OP}$, then
\[
  \lVert \bvec{u} \rVert = \sqrt{x^2 + y^2}
\]
```
↓ Input

↑ Output

Let $\boldsymbol{u} = (x, y)$ represent $\overrightarrow{OP}$, then

$$\lVert \boldsymbol{u} \rVert = \sqrt{x^2 + y^2}$$

↓ Output

Exercise 23 (Maths: Vectors and Arrays)

Try to produce the following:

↑ Output

$$\boldsymbol{Ax} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \end{pmatrix} = \boldsymbol{y}$$

↓ Output

As before, you can download or view the solution.

9.4.12 Mathematical Spacing

LaTeX deals with mathematical spacing fairly well, but sometimes you may find you want to adjust the spacing yourself. Available spacing commands are listed in Table 9.15.

Exercise 24 (More Mathematics)

This exercise uses the spacing command \qquad. In addition, it has a function name, diag, and it uses the \forall and ellipses symbols. It also redefines the \vec command, as was done in the previous section, uses the bmatrix environment (see Section 9.4.10), and has subscripts and superscripts.

Try to reproduce the following output:

Table 9.15 Mathematical Spacing Commands (†provided by amsmath)

Command	**Example Input**	**Example Output**
	`$AB$`	AB
`\thinspace` or `\,`	`$A\,B$`	$A\,B$
`\medspace`† or `\:`	`$A\:B$`	$A\:B$
`\thickspace`† or `\;`	`$A\;B$`	$A\;B$
`\quad`	`$A\quad B$`	$A\quad B$
`\qquad`	`$A\qquad B$`	$A\qquad B$
`\negthinspace` or `\!`	`$A\!B$`	$A\!B$
`\negmedspace`†	`$A\negmedspace B$`	$A\negmedspace B$
`\negthickspace`†	`$A\negthickspace B$`	$A\negthickspace B$

↑ Output

The set of linear equations:

$$a_i x_i = b_i \qquad \forall i = 1, \ldots, n$$

can be written as a matrix equation:

$$\mathrm{diag}(\boldsymbol{a})\boldsymbol{x} = \mathbf{b}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, $\mathbf{b} = (b_1, \ldots, b_n)^T$ and

$$\mathrm{diag}(\boldsymbol{a}) = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix}$$

↓ Output

Again, you can download or view the solution.

Chapter 10

Defining Environments

Just as you can define new commands, you can also define new environments. The command

\newenvironment{⟨*env-name*⟩}[⟨*n-args*⟩][⟨*default*⟩]{⟨*begin-code*⟩}{⟨*end-code*⟩} Definition

is used to define a new environment. As with new commands, you can use the optional argument ⟨*n-args*⟩ to define an environment with arguments, and ⟨*default*⟩ to define an environment with an optional argument.

The first argument ⟨*env-name*⟩ is the name of your new environment. Remember that the environment name must not have a backslash. The mandatory arguments ⟨*begin-code*⟩ and ⟨*end-code*⟩ indicate what LaTeX should do at the beginning and end of the environment. Note that although ⟨*begin-code*⟩ can reference the arguments using #1 etc, the ⟨*end-code*⟩ part can't.

Example (An Exercise Environment):
Let's first consider an example of an environment without any arguments. Let's make an environment called, say, exercise that prints **Exercise** in bold and typesets the contents of the environment in italic, with a gap between the title and the contents. In other words, we want the following code:

↑ Input
```
\begin{exercise}
This is a sample.
\end{exercise}
```
↓ Input

to produce the following output:

↑ Output

Exercise

This is a sample.

↓ Output

(In the next chapter we will add numbering.)

Let's first consider what we want this environment to do: we can get the word "Exercise" in bold using \textbf, and the italic font can be obtained by using the itshape environment (recall Section 4.5). So, at the start of our new environment we need

```
\textbf{Exercise}\begin{itshape}
```
Input

and at the end of our new environment we need to end the itshape environment:

```
\end{itshape}
```

Input

Putting the above together into the new environment definition:

Input

```
\newenvironment{exercise}% environment name
{% begin code
  \textbf{Exercise}\begin{itshape}%
}%
{\end{itshape}}% end code
```

Let's try it out:

Input

```
\begin{exercise}
This is a sample.
\end{exercise}
```

Exercise *This is a sample.*

Output

Not quite right. Let's put a paragraph break after **Exercise**, and put one before it as well. The command `\par` can be used to make a paragraph break and the extra bit of vertical spacing can be produced using `\vspace`. The length `\baselineskip` is the interline spacing. Modifications are shown in bold **`like this`**.

Input

```
\newenvironment{exercise}% environment name
{% begin code
  \par\vspace{\baselineskip}%
  \textbf{Exercise}\begin{itshape}%
  \par\vspace{\baselineskip}%
}%
{\end{itshape}}% end code
```

Let's have a look at the output now:

Output

Exercise

This is a sample.

The indent at the start of each line is caused by the normal paragraph indentation. This can be suppressed using `\noindent`. It's also a good idea to suppress any spaces immediately following `\begin{exercise}` and `\end{exercise}`, which can be done using `\ignorespaces` and `\ignorespacesafterend`. Modifications are again shown in bold **`like this`**.

[FAQ: There's a space added after my environment]