LaTeX/Mathematics

From Wikibooks, the open-content textbooks collection

One of the greatest motivating forces for Donald Knuth when he began developing the original TeX system was to create something that allowed simple construction of mathematical formulas, whilst looking professional when printed. The fact that he succeeded was most probably why TeX (and later on, LaTeX) became so popular within the scientific community. Regardless of the history, typesetting mathematics is one of LaTeX's greatest strengths. It is also a large topic due to the existence of so much mathematical notation.

If you are writing a document that needs only a few simple mathematical formulas, then you can generally use plain LaTeX: it will give you most of the tools you need. However, if you are writing a scientific document that contains numerous complicated formulas, it is highly recommended that you use the `amsmath` package, which introduces several new commands that are more powerful and flexible than the ones provided by plain LaTeX. To use this, include:

```
\usepackage{amsmath}
```

in the preamble of the document.

Mathematics environments

LaTeX needs to know beforehand that the subsequent text does in fact contain mathematical elements. This is because LaTeX typesets maths notation differently than normal text. Therefore, special environments have been declared for this purpose. They can be distinguished into two categories depending on how they are presented:

- *text* - text formulas are displayed in-line, that is, within the body of text where it is declared. e.g., I can say that $a + a = 2a$ within this sentence.
- *displayed* - displayed formulas are separate from the main text.

As maths require special environments, there are naturally the appropriate environment names you can use in the standard way. Unlike most other environments, however, there are some handy shorthands to declaring your formulas. The following table summarizes them:

Type	Environment	LaTeX shorthand	TeX shorthand
Text	`\begin{math}...\end{math}`	`\(...\)`	`$...$`
	`\begin{displaymath}...\end{displaymath}` or		
Displayed	`\begin{equation*}...\end{equation*}`[1]	`\[...\]`	`$$...$$`

Note: Using the `$$...$$` should be avoided, as it may cause problems, particularly with the AMS-LaTeX macros. Furthermore, should a problem occur, the error messages may not be helpful.

Additionally, there is a second possible environment for the *displayed* type of formulas: `equation`. The difference between this and `displaymath` is that `equation` also adds sequential equation numbers by the side.

If you are typing text normally, you are said to be in *text mode*, while you are typing within one of those mathematical environments, you are said to be in *math mode*, that has some differences compared to the *text mode*:

1. Most spaces and line breaks do not have any significance, as all spaces are either derived logically from the mathematical expressions, or have to be specified with special commands such as `\quad`
2. Empty lines are not allowed. Only one paragraph per formula.
3. Each letter is considered to be the name of a variable and will be typeset as such. If you want to typeset normal text within a formula (normal upright font and normal spacing) then you have to enter the text using dedicated commands.

Symbols

Mathematics has lots and lots of symbols! If there is one aspect of maths that is difficult in LaTeX it is trying to remember how to produce them. There are of course a set of symbols that can be accessed directly from the keyboard:

```
+ - = ! / ( ) [ ] < > | ' :
```

Beyond those listed above, distinct commands must be issued in order to display the desired symbols. And there are *a lot!* Greek letters, set and relations symbols, arrows, binary operators, etc. Too many to remember, and in fact, they would overwhelm this tutorial if I tried to list them all. Therefore, for a complete reference document, see the external link at the bottom of the page.

```
\[
 \forall x \in X, \quad \exists y \leq \epsilon
\]
```

$$\forall x \in X, \quad \exists y \leq \epsilon$$

Greek letters

Greek letters are commonly used in mathematics, and they are very easy to type in *math mode*. You just have to type the name of the letter after a backslash: if the first letter is lowercase, you will get a lowercase Greek letter, if the first letter is uppercase (and only the first letter), then you will get an uppercase letter. Note that some uppercase Greek letters look like Latin ones, so they are not provided by LaTeX (e.g. uppercase *Alpha* and *Beta* are just "A" and "B" respectively). Theta, Phi, and Sigma are provided in two different versions:

```
\[
 \alpha, \beta, \gamma, \delta, \epsilon, \zeta, \mu,
 \theta, \vartheta, \phi, \varphi, \omega, \sigma, \varsigma,
 \Gamma, \Delta, \Theta, \Phi, \Omega
\]
```

$$\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \mu, \theta, \vartheta, \phi, \varphi, \omega, \sigma, \varsigma, \Gamma, \Delta, \Theta, \Phi, \Omega$$

Operators

An operator is a function that is written as a word: e.g. trigonometric functions (sin, cos, tan), logarithms and exponentials (log, exp). LaTeX has many of these defined as commands:

```
\[
 \cos (2\theta) = \cos^2 \theta - \sin^2 \theta
\]
```

$$\cos(2\theta) = \cos^2\theta - \sin^2\theta$$

For certain operators such as limits, the subscript is placed underneath the operator:

```
\[
 \lim_{x \to \infty} \exp(-x) = 0
\]
```

$$\lim_{x \to \infty} \exp(-x) = 0$$

For the modular operator there are two commands: `\bmod` and `\pmod`:

```
\[
 a \bmod b
\]
```

$$a \bmod b$$

```
\[
 x \equiv a \pmod b
\]
```

$$x \equiv a \pmod b$$

To use operators which are not pre-defined, such as argmax, see custom operators

Powers and indices

Powers and indices are equivalent to superscripts and subscripts in normal text mode. The caret (^) character is used to raise something, and the underscore (_) is for lowering. If more than one expression is raised or lowered, they should be grouped using curly braces ({ and }).

```
\[
k_{n+1} = n^2 + k_n^2 - k_{n-1}
\]
```

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

Fractions and Binomials

A fraction is created using the `\frac{numerator}{denominator}` command. (For those who need their memories refreshed, that's the *top* and *bottom* respectively!). Likewise, the binomial coefficient (aka the Choose function) may be written using the `\binom` command

```
\[
 \frac{n!}{k!(n-k)!} = \binom{n}{k}
\]
```

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

You can also embed fractions within fractions:

```
\[
 \frac{\frac{1}{x}+\frac{1}{y}}{y-z}
\]
```

$$\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$$

Note that when appearing inside another fraction, or in inline text $\frac{a}{b}$, a fraction is noticeably smaller than in displayed mathematics. The `\tfrac` and `\dfrac` commands[1] force the use of the respective styles (similarly the `\tbinom` and `\dbinom` commands do the same for the binomial coefficient).

For relatively simple fractions, it may be more aesthetically pleasing to use powers and indices:

```
\[
 ^3/_7   
\]
```

$$ {}^3/_7 $$

Roots

The `sqrt` command creates a square root surrounding an expression. It accepts an optional argument specified in square brackets (`[` and `]`) to change magnitude:

```
\[
\sqrt{\frac{a}{b}}
\]
```

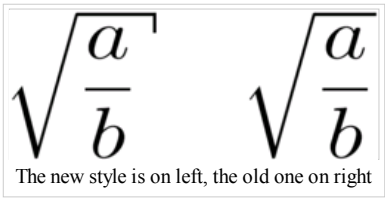
$$ \sqrt{\frac{a}{b}} $$

```
\[
\sqrt[n]{1+x+x^2+x^3+\ldots}
\]
```

$$ \sqrt[n]{1+x+x^2+x^3+\ldots} $$

Some people prefer writing the square root "closing" it over its content. This method arguably makes it more clear just what is in the scope of the root sign. This habit is not normally used while writing with the computer because the text is supposed to be clear anyway, but if you want to change the output of the square root anyway, LaTeX gives you this possibility. Just add the following code in the preamble of your document:

```
% New definition of square root:
% it renames \sqrt as \oldsqrt
\let\oldsqrt\sqrt
% it defines the new \sqrt in terms of the old one
\def\sqrt{\mathpalette\DHLhksqrt}
\def\DHLhksqrt#1#2{%
\setbox0=\hbox{$#1\oldsqrt{#2\,}$}\dimen0=\ht0
\advance\dimen0-0.2\ht0
\setbox2=\hbox{\vrule height\ht0 depth -\dimen0}%
{\box0\lower0.4pt\box2}}
```



This TeX code first renames the `\sqrt` command as `\oldsqrt`, then redefines `\sqrt` in terms of the old one, adding something more. The new square root can be seen in the picture on the right, compared to the old one. Unfortunately this code won't work if you want to use multiple roots: if you try to write $\sqrt[b]{a}$ as `\sqrt[b]{a}` after you used the code above, you'll just get a wrong output. In other words, you can redefine the square root this way only if you are not going to use multiple roots in the whole document.

Sums and integrals

The `\sum` and `\int` commands insert the sum and integral symbols respectively, with limits specified using the caret (`^`) and underscore (`_`):

```
\[
 \sum_{i=1}^{10} t_i
\]
```

$$ \sum_{i=1}^{10} t_i $$

```
\[
 \int_0^\infty e^{-x} \, dx
\]
```

$$ \int_0^\infty e^{-x} \, dx $$

There are many other "big" commands which operate in a similar manner:

`\sum`	$\sum$	`\prod`	$\prod$	`\coprod`	$\coprod$
`\bigoplus`	$\bigoplus$	`\bigotimes`	$\bigotimes$	`\bigodot`	$\bigodot$
`\bigcup`	$\bigcup$	`\bigcap`	$\bigcap$	`\biguplus`	$\biguplus$
`\bigsqcup`	$\bigsqcup$	`\bigvee`	$\bigvee$	`\bigwedge`	$\bigwedge$
`\int`	$\int$	`\oint`	$\oint$	`\iint`[1]	$\iint$
`\iiint`[1]	$\iiint$	`\iiiint`[1]	$\iiiint$	`\idotsint`[1]	$\int\cdots\int$

For more integral symbols, including those not included by default in the Computer Modern font, try the `esint` package. The `\substack` command[1] allows the use of `\\` to write the limits over multiple lines:

```
\[
 \sum_{\substack{
   0<i<m \\
   0<j<n
  }}
 P(i,j)
\]
```

$$\sum_{\substack{0<i<m \\ 0<j<n}} P(i,j)$$

If you want the limits of an integral to be specified above and below the symbol (like the sum), use the `\limits` command:

```
\[
 \int\limits_a^b
\]
```

$$\int\limits_a^b$$

However if you want this to apply to ALL integrals, it is preferable to specify the `intlimits` option when loading the `amsmath` package:

```
\usepackage[intlimits]{amsmath}
```

Brackets, braces and delimiters

The use of delimiters such as brackets soon becomes important when dealing with anything but the most trivial equations. Without them, formulas can become ambiguous. Also, special types of mathematical structures, such as matrices, typically rely on delimiters to enclose them.

There are a variety of delimiters available for use in LaTeX:

```
\[
() \, [] \, \{\} \, || \, \|\| \,
\langle\rangle \, \lfloor\rfloor \, \lceil\rceil
\]
```

$$() \, [] \, \{\} \, || \, \|\| \, \langle\rangle \, \lfloor\rfloor \, \lceil\rceil$$

Automatic sizing

Very often mathematical features will differ in size, in which case the delimiters surrounding the expression should vary accordingly. This can be done automatically using the `left` and `right` commands. Any of the previous delimiters may be used in combination with these:

```
\[
\left(\frac{x^2}{y^3}\right)
\]
```

$$\left(\frac{x^2}{y^3}\right)$$

If a delimiter on only one side of an expression is required, then an invisible delimiter on the other side may be denoted using a period (`.`).

Manual sizing

In certain cases, the sizing produced by the `left` and `right` commands may not be desirable, or you may simply want finer control over the delimiter sizes. In this case, the `big`, `Big`, `bigg` and `Bigg` modifier commands may be used:

```
\[
( \big( \Big( \bigg( \Bigg(
\]
```

$$(\big(\Big(\bigg(\Bigg($$

Matrices and arrays

A basic matrix may be created using the `matrix` environment[1]: in common with other table-like structures, entries are specified by row, with columns separated using an ampersand (`&`) and a new rows separated with a double backslash (`\\`)

```
\[
 \begin{matrix}
  a & b & c \\
  d & e & f \\
  g & h & i
 \end{matrix}
\]
```

$$\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$$

However matrices are usually enclosed in delimiters of some kind, and while it is possible to use the `\left` and `\right` commands, there are various other predefined environments which automatically include delimiters:

Environment name	Surrounding delimiter
`pmatrix`[1]	$( )$

`bmatrix`[1] $\begin{bmatrix}\end{bmatrix}$

`Bmatrix`[1] $\begin{Bmatrix}\end{Bmatrix}$

`vmatrix`[1] $\begin{vmatrix}\end{vmatrix}$

`Vmatrix`[1] $\begin{Vmatrix}\end{Vmatrix}$

When writing down arbitrary sized matrices, it is common to use horizontal, vertical and diagonal triplets of dots (known as ellipses) to fill in certain columns and rows. These can be specified using the `\cdots`, `\vdots` and `\ddots` respectively:

```
\[
 A_{m,n} =
 \begin{pmatrix}
  a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
  a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  a_{m,1} & a_{m,2} & \cdots & a_{m,n}
 \end{pmatrix}
\]
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

In some cases you may want to have finer control of the alignment within each column, or want to insert lines between columns or rows. This can be achieved using the `array` environment, which is essentially a math-mode version of the `tabular` environment, which requires that the columns be pre-specified:

```
\[
 \begin{array}{c|c}
  1 & 2 \\
  \hline
  3 & 4
 \end{array}
\]
```

$$\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}$$

Adding text to equations

The math environment differs from the text environment in the representation of text. Here is an example of trying to represent text within the math environment:

```
\[
 50 apples \times 100 apples = lots of apples^2
\]
```

$$50apples \times 100apples = lotsofapples^2$$

There are two noticeable problems: there are no spaces between words or numbers, and the letters are italicized and more spaced out than normal. Both issues are simply artifacts of the maths mode, in that it treats it as a mathematical expression: spaces are ignored (LaTeX spaces mathematics according to its own rules), and each character is a separate element (so are not positioned as closely as normal text).

There are a number of ways that text can be added properly. The typical way is to wrap the text with the `\text{...}` command [1] (a similar command is `\mbox{...}`, though this causes problems with subscripts, and has a less descriptive name). Let's see what happens when the above equation code is adapted:

```
\[
 50 \text{apples} \times 100 \text{apples} = \text{lots of apples}^2
\]
```

$$50\text{apples} \times 100\text{apples} = \text{lots of apples}^2$$

The text looks better. However, there are no gaps between the numbers and the words. Unfortunately, you are required to explicitly add these. There are many ways to add spaces between maths elements, but for the sake of simplicity you may literally add the space character in the affected `\text`(s) itself (just before the text.)

```
\[
 50 \text{ apples} \times 100 \text{ apples} = \text{lots of apples}^2
\]
```

$$50 \text{ apples} \times 100 \text{ apples} = \text{lots of apples}^2$$

Formatted text

Using the `\text` is fine and gets the basic result. Yet, there is an alternative that offers a little more flexibility. You may recall the introduction of font formatting commands, such as `\textrm`, `\textit`, `\textbf`, etc. These commands format the argument accordingly, e.g., `\textbf{bold text}` gives **bold text**. These commands are equally valid within a maths environment to include text. The added benefit here is that you can have better control over the font formatting, rather than the standard text achieved with `\text`.

```
\[
 50 \textrm{ apples} \times 100 \textbf{ apples} = \textit{lots of apples}^2
\]
```

$$50 \text{ apples} \times 100 \textbf{ apples} = \textit{lots of apples}^2$$

Formatting mathematics symbols

So we can format text, what about formatting mathematics? There are a set of formatting commands very similar to the font formatting ones just used, except they are aimed specifically for text in maths mode:

LaTeX command	Sample	Description	Common use
`\mathnormal{…}`	$ABCDEFabcdef123456$	the default math font	most mathematical notation
`\mathrm{…}`	ABCDEFabcdef123456	this is the default or normal font, unitalicised	units of measurement, one word functions
`\mathit{…}`	$ABCDEFabcdef123456$	italicised font	
`\mathbf{…}`	**ABCDEFabcdef123456**	bold font	vectors
`\mathsf{…}`	ABCDEFabcdef123456	Sans-serif	
`\mathtt{…}`	ABCDEFabcdef123456	Monospace (fixed-width) font	
`\mathcal{…}`	$\mathcal{ABCDEF}\dashv\sqcup\sqcap\{\infty\in\ni\triangle\triangledown/$	Calligraphy (uppercase only)	often used for sheaves/schemes and categories
`\mathfrak{…}`[2]	$\mathfrak{ABCDEFabcdef123456}$	Fraktur	Almost canonical font for Lie algebras
`\mathbb{…}`[2]	$\mathbb{ABCDEF}\supset\cup\nVdash\nvDash\nvdash\ntrianglerighteq\ntriangleleft$	Blackboard bold	Used to denote special sets (e.g. real numbers)
`\mathscr{…}`[3]		Script	

The maths formatting commands can be wrapped around the entire equation, and not just on the textual elements: they only format letters, numbers, and uppercase Greek, and the rest of the maths syntax is ignored.

To bold lowercase Greek or other symbols use the `\boldsymbol` command[1]; this will only work if there exists a bold version of the symbol in the current font. As a last resort there is the `\pmb` command[1] (poor mans bold): this prints multiple versions of the character slightly offset against each other

```
\[
 \boldsymbol{\beta} = (\beta_1,\beta_2,\ldots,\beta_n)
\]
```

$$\boldsymbol{\beta} = (\beta_1,\beta_2,\ldots,\beta_n)$$

To change the size of the fonts in math mode, see Changing font size.

Accents

So what to do when you run out of symbols and fonts? Well the next step is to use accents:

`a'`	a'	`a''`	a''	`a'''`	a'''	`a''''`	a''''		
`\hat{a}`	$\hat{a}$	`\bar{a}`	$\bar{a}$	`\overline{aaa}`	$\overline{aaa}$	`\check{a}`	$\check{a}$	`\tilde{a}`	$\tilde{a}$
`\grave{a}`	$\grave{a}$	`\acute{a}`	$\acute{a}$	`\breve{a}`	$\breve{a}$	`\vec{a}`	$\vec{a}$		
`\dot{a}`	$\dot{a}$	`\ddot{a}`	$\ddot{a}$	`\dddot{a}`[1]		`\ddddot{a}`[1]			
`\not{a}`	$\not{a}$	`\mathring{a}`		`\widehat{AAA}`	$\widehat{AAA}$	`\widetilde{AAA}`			

Plus and minus signs

Latex deals with the + and − signs in two possible ways. The most common is as a binary operator. When two maths elements appear either side of the sign, it is assumed to be a binary operator, and as such, allocates some space either side of the sign. The alternative way is a sign designation. This is when you state whether a mathematical quantity is either positive or negative. This is common for the latter, as in maths, such elements are assumed to be positive unless a - is prefixed to it. In this instance, you want the sign to appear close to the appropriate element to show their association. If you put a + or a - with nothing before it but you want it to be handled like a binary operator you can add an *invisible* character before the operator using `{}`. This can be useful if you are writing multiple-line formulas, and a new line could start with a = or a +, for example, then you can fix some strange alignments adding the invisible character where necessary.

Controlling horizontal spacing

Latex is obviously pretty good at typesetting maths—it was one of the chief aims of the core Tex system that Latex extends. However, it can't always be relied upon to accurately interpret formulas in the way you did. It has to make certain assumptions when there are ambiguous expressions. The result tends to be slightly incorrect horizontal spacing. In these events, the output is still satisfactory, yet, any perfectionists will no doubt wish to *fine-tune* their formulas to ensure spacing is correct. These are generally very subtle adjustments.

There are other occasions where Latex has done its job correctly, but you just want to add some space, maybe to add a comment of some kind. For example, in the following equation, it is preferable to ensure there is a decent amount of space between the maths and the text.

```
\[
f(n) = \left\{
\begin{array}{l l}
  n/2 & \quad \text{if $n$ is even}\\
  -(n+1)/2 & \quad \text{if $n$ is odd}\\
\end{array} \right.
\]
```

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ -(n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

Latex has defined two commands that can be used anywhere in documents (not just maths) to insert some horizontal space. They are `\quad` and `\qquad`

A `\quad` is a space equal to the current font size. So, if you are using an 11pt font, then the space provided by `\quad` will also be 11pt (horizontally, of course.) The `\qquad` gives twice that amount. As you can see from the code from the above example, `\quad`s were used to add some separation between the maths and the text.

OK, so back to the fine tuning as mentioned at the beginning of the document. A good example would be displaying the simple equation for the indefinite integral of *y* with respect to *x*:

$$\int y \, \mathrm{d}x$$

If you were to try this, you may write:

```
\[ \int y \mathrm{d}x \]
```

$$\int y \mathrm{d}x$$

However, this doesn't give the correct result. Latex doesn't respect the white-space left in the code to signify that the *y* and the d*x* are independent entities. Instead, it lumps them altogether. A `\quad` would clearly be overkill is this situation—what is needed are some small spaces to be utilized in this type of instance, and that's what Latex provides:

Command	Description	Size
`\,`	small space	3/18 of a quad
`\:`	medium space	4/18 of a quad
`\;`	large space	5/18 of a quad
`\!`	negative space	-3/18 of a quad

NB you can use more than one command in a sequence to achieve a greater space if necessary.

So, to rectify the current problem:

```
\int y\, \mathrm{d}x
```

$$\int y \, \mathrm{d}x$$

```
\int y\: \mathrm{d}x
```

$$\int y \: \mathrm{d}x$$

```
\int y\; \mathrm{d}x
```

$$\int y \; \mathrm{d}x$$

The negative space may seem like an odd thing to use, however, it wouldn't be there if it didn't have *some* use! Take the following example:

```
\[\left(
  \begin{array}{c}
    n \\
    r
  \end{array}
  \right) = \frac{n!}{r!(n-r)!}
\]
```

$$\left(\begin{array}{c} n \\ r \end{array} \right) = \frac{n!}{r!(n-r)!}$$

The matrix-like expression for representing binomial coefficients is too padded. There is too much space between the brackets and the actual contents within. This can easily be corrected by adding a few negative spaces after the left bracket and before the right bracket.

```
\[\left(\!
  \begin{array}{c}
    n \\
    r
  \end{array}
  \!\right) = \frac{n!}{r!(n-r)!}
\]
```

$$\left(\!\begin{array}{c} n \\ r \end{array}\!\right) = \frac{n!}{r!(n-r)!}$$

In any case, adding some spaces manually should be avoided whenever possible: it makes the source code more complex and it's against the basic principles of a What You See is What You Mean approach. The best thing to do is to define some commands using all the spaces you want and then, when you use your command, you don't have to add any other space. Later, if you change your mind about the length of the horizontal space, you can easily change it modifying only the command you defined before. Let us use an example: you want the *d* of a *dx* in an integral to be in roman font and a small space away from the rest. If you want to type an integral like `\int x \; \mathrm{d} x`, you can define a command like this:

```
\newcommand{\dd}{\; \mathrm{d}}
```

in the preamble of your document. We have chosen `\dd` just because it reminds the "d" it replaces and it is fast to type. Doing so, the code for your integral becomes `\int x \dd x`. Now, whenever you write an integral, you just have to use the `\dd` instead of the "d", and all your integrals will have the same style. If you change your mind, you just have to change the definition in the preamble, and all your integrals will be changed accordingly.

Advanced Mathematics: AMS Math package

The AMS (American Mathematical Society) mathematics package is a powerful package that creates an higher layer of abstraction over mathematical LaTeX language; if you use it it will make your life easier. Some commands amsmath introduces will make other plain LaTeX commands obsolete: in order to keep consistency in the final output you'd better use `amsmath` commands whenever possible. If you do so, you will get an elegant output without worrying about alignment and other details, keeping your source code readable. If you want to use it, you have to add this in the preamble:

```
\usepackage{amsmath}
```

Introducing text and dots in formulas

Amsmath defines also the `\dots` command, that is a generalization of the existing `\ldots`. You can use `\dots` in both text and math mode and LaTeX will replace it with three dots "…" but it will decide according to the context whether to put it on the bottom (like `\ldots`) or centered (like `\cdots`).

Dots

LaTeX gives you several commands to insert dots in your formulas. This can be particularly useful if you have to type big matrices omitting elements. First of all, here are the main dots-related commands LaTeX provides:

Code	Output	Comment
`\dots`	$\dots$	generic dots, to be used in text (outside formulas as well). It automatically manages whitespaces before and after itself according to the context, it's a higher level command.
`\ldots`	$\ldots$	the output is similar to the previous one, but there is no automatic whitespace management; it works at a lower level.
`\cdots`	$\cdots$	These dots are centered relative to the height of a letter. There is also the binary multiplication operator, `\cdot`, mentioned below.
`\vdots`	$\vdots$	vertical dots
`\ddots`	$\ddots$	diagonal dots
`\iddots`		inverse diagonal dots (requires the mathdots package)
`\hdotsfor{n}`	$\cdots\cdots$	to be used in matrices, it creates a row of dots spanning *n* columns.

List of Mathematical Symbols

All the pre-defined mathematical symbols from the \TeX\ package are listed below. More symbols are available from extra packages.

Relation Symbols

Symbol	Script	Symbol	Script	Symbol	Script	Symbol	Script	Symbol	Script
$\leq$	`\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`	$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`
$\neq$	`\neq`	$\smile$	`\smile`	$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\doteqdot$	`\doteq`
$\frown$	`\frown`	$\in$	`\in`	$\ni$	`\ni`	$\propto$	`\propto`	$=$	`=`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$<$	`<`	$>$	`>`		

Greek Letters

Symbol	Script
A and α	`\Alpha` and `\alpha`
B and β	`\Beta` and `\beta`
Γ and γ	`\Gamma` and `\gamma`
Δ and δ	`\Delta` and `\delta`
E, ϵ and ε	`\Epsilon`, `\epsilon` and `\varepsilon`
Z and ζ	`\Zeta` and `\zeta`
H and η	`\Eta` and `\eta`
Θ, θ and ϑ	`\Theta`, `\theta` and `\vartheta`
I and ι	`\Iota` and `\iota`
K and κ	`\Kappa` and `\kappa`
Λ and λ	`\Lambda` and `\lambda`
M and μ	`\Mu` and `\mu`
N and ν	`\Nu` and `\nu`
Ξ and ξ	`\Xi` and `\xi`
Π, π and ϖ	`\Pi`, `\pi` and `\varpi`
P, ρ and ϱ	`\Rho`, `\rho` and `\varrho`
Σ, σ and ς	`\Sigma`, `\sigma` and `\varsigma`
T and τ	`\Tau` and `\tau`
Υ and υ	`\Upsilon` and `\upsilon`
Φ, ϕ, and φ	`\Phi`, `\phi` and `\varphi`
X and χ	`\Chi` and `\chi`
Ψ and ψ	`\Psi` and `\psi`
Ω and ω	`\Omega` and `\omega`

Binary Operations

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

Set and/or Logic Notation

Symbol	Script
$\exists$	`\exists`
$\forall$	`\forall`
$\neg$	`\neg`
$\in$ and $\notin$	`\in` and `\notin`
$\ni$	`\ni`
$\land$	`\land`
$\lor$	`\lor`
$\implies$	`\implies`
$\iff$	`\iff`
$\top$	`\top`
$\bot$	`\bot`
$\emptyset$	`\emptyset`

Summary

As you can begin to see, typesetting maths can be tricky at times. However, because Latex provides so much control, you can get professional quality mathematics typesetting for relatively little effort (once you've had a bit of practice, of course!). It would be possible to keep going and going with maths topics because it seems potentially limitless. However, with this tutorial, you should be able to get along sufficiently.

Notes

1. ↑ $\mathcal{a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r}$ requires the `amsmath` package
2. ↑ $\mathfrak{a\ b}$ requires `amsfonts` or `amssymb` packages
3. ↑ require `mathrsfs` package

Further reading

- meta:Help:Displaying a formula: Wikimedia uses a subset of LaTeX commands.

External links

- LaTeX maths symbols (http://www.artofproblemsolving.com/Wiki/index.php/LaTeX:Symbols)
- detexify (http://detexify.kirelabs.org) : applet for looking up LaTeX symbols by handwriting them
- `amsmath` documentation (ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf)
- LaTeX - The Student Room (http://www.thestudentroom.co.uk/wiki/LaTeX)