Chapter 13

Mathematics

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. However, 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 all of the tools you need. However, if you are writing a scientific document that contains numerous complicated formulas, then you'll most likely need to use the `amsmath` package. It introduces several new commands that are more powerful and easy-to-use than the ones provided by plain LaTeX.

Basic Mathematics: plain LaTeX

All the commands discussed in this section can be used in LaTeX without loading any external package. What is here is enough if you just want to write a few formulas, otherwise you'd better read the advanced section as well. In any case, this is a necessary introduction to how LaTeX can manage mathematics.

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}`	`\(...\)`	`$...$`
Displayed	`\begin{displaymath}... \end{displaymath}`	`\[...\]`	`$$...$$`

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.

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 and Phi are provided in two different versions:

```
\[
\alpha, \beta, \gamma, \mu,
\theta, \vartheta, \phi, \varphi, \omega,
\Gamma, \Theta, \Phi, \Omega
\]
```

$$\alpha, \beta, \gamma, \mu, \theta, \vartheta, \phi, \varphi, \omega, \Gamma, \Theta, \Phi, \Omega$$

Set letters

Set letters are commonly used in Maths to name special sets like the set of natural numbers. To create one, just type `\mathbb{''letter''}` and put the letter in the brackets (using the `amssymb` package). It works only with capitals.

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

$$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$$

If you prefer bold letters, so choose `\mathbf{letter}`.

```
\[
\mathbf{N}, \mathbf{Z}, \mathbf{Q},
\mathbf{R}, \mathbf{C}
\]
```

$$\mathbf{N}, \mathbf{Z}, \mathbf{Q}, \mathbf{R}, \mathbf{C}$$

Note that you cannot bold greek letter using `\mathbf`. The `bm` package defines the `\bm` command which can do this.

Fractions

To create a fraction, you must use the `\frac{numerator}{denominator}` command. (For those who need their memories refreshed, that's the *top* and *bottom* respectively!)

You can also embed fractions within fractions, as shown in the examples below:

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

It is also possible to produce a simple fraction of the form

$$^x/_y$$

using the command:

```
^x/_y
```

Powers and indices

Powers and indices are mathematically equivalent to superscripts and subscripts in normal text mode. The carat (ˆ) character is used to raise something, and the underscore (_) is for lowering. How to use them is best shown by example:

Power		**Index**	
`x^n`	x^n	`n_i`	n_i
`x^{2n}`	x^{2n}	`n_{ij}`	n_{ij}

Note: if more than one character is to be raised (or lowered) then you must group them using the curly braces ({ and }).

Also, if you need to assign both a power and an index to the same entity, then that is achieved like this: `x^{2i}_{3j}` (or `x_{3j}^{2i}`, order is not significant).

$$x^{2i}_{3j}$$

Roots

Typically, for the majority of times, you are after the square root, which is done easily using the following command: `\sqrt{x}`. However, this can be generalized to produce a root of any magnitude:

`\sqrt[n]{x}`	$\sqrt[n]{x}$

Latex will automatically ensure that the size of the root notation adjusts to the size of the contents.

The n is optional, and without it will output a square root. Also, regardless of the size of root you're after, e.g., n=3, you still use the `\sqrt` command.

See also Tips and Tricks for another look of root.

Brackets

The use of 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 brackets to enclose them.

You may recall that you have the `( ) [ ] { }` | | symbols at your disposal, curly brackets requiring a prefixed backslash. Larger structures sometimes require a specified representation. This is shown by example:

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

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

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

The first example shows what would happen if you used the standard bracket characters. As you can see, they would be fine for a simple equation that remained on a single line (e.g., (3 + 2) x (10-3) = 35) but not for equations that have greater vertical size, such as those using fractions. The second example illustrates the LaTeX way of coping with this problem.

The \`left#` and \`right#` commands provide the means for automatic sizing of brackets, curly braces or absolute value symbols. You must enclose the expression that you want in brackets (or absolute values) with these commands. The # after the command should be replaced with the style of bracket desired.

A problem happens if you want to split the equation across multiple lines, as the right bracket can only be used if there is a remaining open left bracket on the current line (though the types do not need to match). If you need to force a right bracket you can make an invisible left bracket and a full stop: \`left.`

Alternatively, check out the `nath` package, which provides auto-scaling delimiters.

Arrays

Using the `array` environment you can create table-like structures in *math mode*. The `array` environment is basically equivalent to the `tabular` environment. For example, if you want to create a matrix, Latex, by default, doesn't have a specific command to use, but you can create a similar structure using `array`. You can use the array to arrange and align your data as you want, and then enclose it with appropriate left and right brackets, and this will give you your matrix. For a simple 2x2 matrix:

```
\[ \left[
  \begin{array}{ c c }
     1 & 2 \\
     3 & 4
  \end{array} \right]
\]
```

$$\left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}\right]$$

Arrays are very flexible; here is an example of another matrix-like structure:

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

$$\left(\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array} \right)$$

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:

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

$50 apples \times 100 apples = lots of apples^2$

There are two noticeable problems. Firstly, there are no spaces between numbers and text, nor spaces between multiple words. Secondly, the words don't look quite right—the letters are more spaced out than normal. Both issues are simply artifacts of the maths mode, in that it doesn't expect to see words. Any spaces that you type in maths mode are ignored and Latex spaces elements according to its own rules. It is assumed that any characters represent variable names. To emphasize that each symbol is an individual, they are not positioned as closely together as with normal text.

There are a number of ways that text can be added properly. The typical way is to wrap the text with the `\mbox{...}` command. This command hasn't been introduced before, however, its job is basically to create a text box just wide enough to contain the supplied text. Text within this box cannot be broken across lines. Let's see what happens when the above equation code is adapted:

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

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, however, for the sake of simplicity, I find it easier, in this instance at least, just to literally add the space character in the affected `\mbox`(s) itself (just before the text.)

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

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

Formatted text

Using the `\mbox` 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 `\mbox`.

```
\begin{equation}
  50 \textrm{ apples} \times 100 \textbf{ apples} =
  \textit{lots of apples}^2
\end{equation}
```

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

However, as is the case with Latex, there is more than one way to skin a cat! 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. So why bother showing you `\textrm` and co if there are equivalents for maths? Well, that's because they are subtly different. The maths formatting commands are:

LaTeX command	Sample	Description	Common use
`\mathnormal{...}`	$\mathnormal{ABCDEFabcdef123456}$	the default math font	most mathematical notation
`\mathrm{...}`	$\mathrm{ABCDEFabcdef123456}$	this is the default or normal font, unitalicised	units of measurement, one word functions
`\mathit{...}`	$\mathit{ABCDEFabcdef123456}$	italicised font	
`\mathbf{...}`	$\mathbf{ABCDEFabcdef123456}$	bold font	vectors
`\mathsf{...}`	$\mathsf{ABCDEFabcdef123456}$	Sans-serif	
`\mathtt{...}`	$\mathtt{ABCDEFabcdef123456}$	Monospace (fixed-width) font	
`\mathcal{...}`	$\mathcal{ABCDEF}\dashv\lfloor\rfloor\lceil\rceil\{\infty\in\ni\triangle\bigtriangledown/$	calligraphy	often used for sheaves/schemes and categories
`\mathfrak{...}`[1]	$\mathfrak{ABCDEFabcdef123456}$	Fraktur	Almost canonical font for Lie algebras
`\mathbb{...}`[1]	$\mathbb{ABCDEF}\Game\mho\nVdash\nvDash\nVDash\ntrianglerighteq\ntrianglelefteq\ntriangleleft$	Blackboard bold	Used to denote special sets (e.g. real numbers)
`\mathscr{...}`[2]		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

[1]requires `amsfonts` or `amssymb` packages
[2]require `mathrsfs` package

Greek, and the rest of the maths syntax is ignored. So, generally, it is better to use the specific maths commands if required. Note that the calligraphy example gives rather strange output. This is because for letters, it requires upper case characters. The remaining letters are mapped to special symbols.

Changing text size of equations

Probably a rare event, but there may be a time when you would prefer to have some control of the size. For example, using text-mode maths, by default a simple fraction will look like this: $\frac{a}{b}$ where as you may prefer to have it displayed larger, like when in display mode, but still keeping it in-line, like this

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

.

A simple approach is to utilize the predefined sizes for maths elements:

`\displaystyle`	Size for equations in display mode
`\textstyle`	Size for equations in text mode
`\scriptstyle`	Size for first sub/superscripts
`\scriptscriptstyle`	Size for subsequent sub/superscripts

A classic example to see this in use is typesetting continued fractions. The following code provides an example.

```
\begin{equation}
  x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + a_4}}}
\end{equation}
```

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + a_4}}}$$

As you can see, as the fractions continue, they get smaller (although they will not get any smaller as in this example, they have reached the `\scriptstyle` limit. If you wanted to keep the size consistent, you could declare each fraction to use the display style instead, e.g.:

```
\begin{equation}
  x = a_0 + \frac{1}{\displaystyle a_1
          + \frac{1}{\displaystyle a_2
          + \frac{1}{\displaystyle a_3 + a_4}}}
\end{equation}
```

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + a_4}}}$$

Another approach is to use the `\DeclareMathSizes` command to select your preferred sizes. You can only define sizes for `\displaystyle`, `\textstyle`, etc. One potential downside is that this command sets the global maths sizes, as it can only be used in the document preamble.

However, it's fairly easy to use: `\DeclareMathSizes{ds}{ts}{ss}{sss}`, where *ds* is the *display size*, *ts* is the *text size*, etc. The values you input are assumed to be point (pt) size.

NB the changes only take place if the value in the first argument matches the current document text size. It is therefore common to see a set of declarations in the preamble, in the event of the main font being changed. E.g.,

```
\DeclareMathSizes{10}{18}{12}{8}   % For size 10 text
\DeclareMathSizes{11}{19}{13}{9}   % For size 11 text
\DeclareMathSizes{12}{20}{14}{10}  % For size 12 text
```

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.

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

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

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 dx 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.

Argmax and argmin

LaTeX has no built-in "\argmax" command to typeset argmax. Some people get around this by using **\arg\max**. This could be undesirable, because a subscripted variable will appear centered beneath the word "max", instead of centered beneath the whole word.

The following command can be used to correctly display the argmax operator:

```
\underset{x}{\operatorname{arg\,max}}
```

$$\underset{x}{\operatorname{arg\,max}}$$

Another way is to define the command:

```
\newcommand{\argmax}{\operatornamewithlimits{arg\,max}}
```

$$\underset{x}{\operatorname{arg\,max}}$$

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

You have been told to use `\mbox{...}` to insert text within formulas. Amsmath provides another command for it, that is `\text{...}`. It works like `\mbox`, but it's better because it will adjust the size of the text according to the context. For example, if you want to write text in a subscript, if you use `\mbox` the text will remain big, if you use `\text` the text will look smaller, as you would expect.

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`).

`\dot{x}`	$\dot{x}$
`\ddot{x}`	$\ddot{x}$
`\hat{x}`	$\hat{x}$
`\widehat{x}`	$\widehat{x}$

Showing formulas

If you want to write a formula within the text, you still have to use `$ ... $` but, if you want to write a big formula on its own line, then amsmath introduces lots of changes. In the following sections there are all the possible ways to insert an in-line formula, with a description on how to do it. Any possible output can be expressed in terms of the following structures. Do not use the environments introduced in the plain LaTeX section: they have a different management of spaces before and after the formula, so you'd better use only amsmath environments for consistency. Moreover, amsmath prevents overlapping of the equation numbers with wide formula, while plain LaTeX does not.

One centered formula, without any label

When you just want to show a formula without referencing it later, use the `equation*` environment. Here is an example:

```
\begin{equation*}
a x^2 + b x + c = 0
\end{equation*}
```

$$ax^2 + bx + c = 0$$

One centered formula, with label

When you just want to show a formula and you want to reference it later, use the `equation` environment. Here is an example

```
\begin{equation}
a x^2 + b x + c = 0
\end{equation}
```

$$ax^2 + bx + c = 0 \tag{1}$$

Several centered formulas, without label

When you want to show several centered formulas without referencing them later use `gather*`. It can be useful if you want to show the steps leading to a conclusion:

```
\begin{gather*}
a x + b = 0 \\
a x^2 + b x + c = 0 \\
a x^3 + b x^2 + c x + d = 0
\end{gather*}
```

$$\begin{gathered} ax + b = 0 \\ ax^2 + bx + c = 0 \\ ax^3 + bx^2 + cx + d = 0 \end{gathered}$$

Several centered formulas, one label for all of them

When you want to show several formulas on different lines, but you want to reference all of them just with one number, then you have to use the `gathered` environment within `equation`. Here is an example:

```
\begin{equation}
\begin{gathered}
a x + b = 0 \\
a x^2 + b x + c = 0 \\
a x^3 + b x^2 + c x + d = 0
\end{gathered}
\end{equation}
```

$$\begin{gathered} ax + b = 0 \\ ax^2 + bx + c = 0 \\ ax^3 + bx^2 + cx + d = 0 \end{gathered} \tag{2}$$

Several centered formulas, each with its own label

When you want to show several formulas on different lines, but you want to be able to reference each of them by a different number, you have to use `gather`. Here is an example:

```
\begin{gather}
a x + b = 0 \\
a x^2 + b x + c = 0 \\
a x^3 + b x^2 + c x + d = 0
\end{gather}
```

$$ax + b = 0 \tag{3}$$
$$ax^2 + bx + c = 0 \tag{4}$$
$$ax^3 + bx^2 + cx + d = 0 \tag{5}$$

If you want to number all the lines but one or two, you can add the command `\notag` on the line you want not to be numbered.

Several formulas, any alignment, without label

When you want to show one or several formulas with a particular alignment you want to define, you have to use `flalign*`. Adding a \\ you start a new line, using & you can manage the alignment. On each line you can add as many `&` as you want, but there must be the same number of any line, otherwise LaTeX returns an error. On each line, the alignment is managed as following:

```
\begin{flalign*}
right & left & right & left & right & left \\
right & left & right & left & right & left
\end{flalign*}
```

you can add multiple & without anything between them, for example:

```
\begin{flalign*}
& left & & left & & left \\
& left & & left & & left
\end{flalign*}
```

it's quite hard to center a formula using `flalign*`, but it doesn't matter since you can use all the other environments defined above if you want to. Here is a practical example:

```
\begin{flalign*}
10xy^2+15x^2y-5xy & =  5\left(2xy^2+3x^2y-xy\right) = \\
   & = 5x\left(2y^2+3xy-y\right) = \\
   & = 5xy\left(2y+3x-1\right)
\end{flalign*}
```

$$\begin{aligned}10xy^2+15x^2y-5xy &= 5\left(2xy^2+3x^2y-xy\right) = \\ &= 5x\left(2y^2+3xy-y\right) = \\ &= 5xy\left(2y+3x-1\right)\end{aligned}$$

Several formulas, any alignment, each with its own label

If you want to align your formulas as you want, but you want each line to be numbered, just use `flalign`. It works exactly like the starred version. Here is an example:

```
\begin{flalign}
10xy^2+15x^2y-5xy & =  5\left(2xy^2+3x^2y-xy\right) = \\
   & = 5x\left(2y^2+3xy-y\right) = \\
   & = 5xy\left(2y+3x-1\right)
\end{flalign}
```

$$\begin{aligned}10xy^2+15x^2y-5xy &= 5\left(2xy^2+3x^2y-xy\right) = && (6)\\ &= 5x\left(2y^2+3xy-y\right) = && (7)\\ &= 5xy\left(2y+3x-1\right) && (8)\end{aligned}$$

Several formulas, any alignment, one label for all of them

If you want to manage the alignment as you like, but you want to be able to reference all the lines just with one number, then you have to use `split` within `equation`. Here is an example:

```
\begin{equation}
\begin{split}
10xy^2+15x^2y-5xy & =  5\left(2xy^2+3x^2y-xy\right) = \\
   & = 5x\left(2y^2+3xy-y\right) = \\
   & = 5xy\left(2y+3x-1\right)
\end{split}
\end{equation}
```

$$\begin{split}10xy^2+15x^2y-5xy &= 5\left(2xy^2+3x^2y-xy\right) = \\ &= 5x\left(2y^2+3xy-y\right) = \\ &= 5xy\left(2y+3x-1\right)\end{split} \qquad (9)$$

Splitting long formulas

LaTeX does not take care of splitting long formulas in several lines, you have to do it by yourself. One possible approach is to separate the long formula into smaller parts and align it manually; this way you will get the best approach according to your needs. Anyway, if you want LaTeX to work for you, you'd better use the `multline` environment. If you use it, all you have to do is to add \\ where you want to start a new line. LaTeX will set the alignment automatically: the first line will be left-aligned, the last line right-aligned, all the lines in the middle will be centered. Here is an example:

```
\begin{multline}
\left(1+x\right)^n  = 1 + nx + \frac{n\left(n-1\right)}{2!}x^2 +\\
+ \frac{n\left(n-1\right)\left(n-2\right)}{3!}x^3 +\\
+ \frac{n\left(n-1\right)\left(n-2\right)\left(n-3\right)}{4!}x^4 + \dots
\end{multline}
```

$$\begin{multline}\left(1+x\right)^n = 1 + nx + \frac{n\left(n-1\right)}{2!}x^2 +\\ + \frac{n\left(n-1\right)\left(n-2\right)}{3!}x^3 +\\ + \frac{n\left(n-1\right)\left(n-2\right)\left(n-3\right)}{4!}x^4 + \dots \tag{10}\end{multline}$$

There is a starred version `multline*` if you don't want to label it.

Remember that, if you are using adapting brackets such as `\left[ ... \right]` you cannot start a new line until all those brackets have been closed. The way to fix this is to put a period (the invisible delimiter) to match – using e.g. `\left( ... \right.`. To get the heights right, add an invisible object with height, using e.g. `\vphantom{\frac{1}{2}}`. Another way to take care of this is to set the size yourself with, for instance, one of `\big( \Big( \bigg( \Bigg(`.

Other options

If you want several formulas to be labeled with the same number, but you still want to identify them with different letters, then you can use the `subequations` environments. You put `\begin{subequations}` outside any mathematical environment; until LaTeX finds `\end{subequations}`, all the labels for equations will have the same number with a letter next to it. For example, they'll be labeled 11a, 11b, 11c, etc. Here is an example:

```
\begin{subequations}
\begin{gather}
a x + b = 0 \\
a x^2 + b x + c = 0 \\
a x^3 + b x^2 + c x + d = 0
\end{gather}
\end{subequations}
```

$$ax + b = 0 \tag{11a}$$
$$ax^2 + bx + c = 0 \tag{11b}$$
$$ax^3 + bx^2 + cx + d = 0 \tag{11c}$$

you can use any mathematical environment within `subequations`.

If you want to underline the importance of a formula, you can put it inside a box. To do so, choose the mathematical environment you prefer and type your formula as an argument of `\boxed{...}`. Here is an example:

```
\begin{equation*}
\boxed{a x^2 + b x + c = 0}
\end{equation*}
```

$$\boxed{ax^2 + bx + c = 0}$$

Matrix-like environments

Amsmath introduces several commands that will help you typing matrices. In plain TeX, the only way is to use the `array` environment, but you have to define how many columns you want and how you want them aligned, just like a table. If you use the environments amsmath introduces, you don't have to worry about it anymore, LaTeX will take care of it. Those environments are:

```
\begin{environment}
  a & b \\
  c & d
\end{environment}
```

environment	output
`matrix`	$\begin{matrix} a & b \\ c & d \end{matrix}$
`pmatrix`	$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$
`bmatrix`	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
`Bmatrix`	$\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$
`vmatrix`	$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$
`Vmatrix`	$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$

As you can see, the syntax within the environment is just like `array`. Another environment introduced by amsmath that is based on `array` is the `cases` environment, that you can use to write "cases":

```
u(x) =
\begin{cases}
1 & \text{if } x \geq 0 \\
0 & \text{if } x < 0
\end{cases}
```

$$u(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Just like before, you don't have to take care of definition or alignment of columns, LaTeX will do it for you.

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`	$\cdots$	those dots are centered relative to the height of a letter
`\vdots`	$\vdots$	vertical dots
`\ddots`	$\ddots$	diagonal dots
`\hdotsfor{''n''}`	$\dots\dots$	to be used in matrices, it creates a row of dots spanning n columns.

Using those commands it is possible to create any complicated output with a simple and elegant code. Here is a practical example:

```
\begin{equation}
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}
\end{equation}
```

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

Integrals and sums

Amsmath introduces more symbols for integrals:

`\int`	$\int$
`\iint`	$\iint$
`\iiint`	$\iiint$
`\iiiint`	$\iiiint$
`\idotsint`	$\displaystyle\int\cdots\int$

this way the multiple integrals will look closer than simply using `\int` several times. Moreover, if you add subscripts to an integral, in standard LaTeX they will look like $\int_a^b$ (`\int_{a}^{b}`). If you want a and b to be placed on the bottom and top of those symbols (this is the behavior you would expect using limits), you have to load the package with `intlimits`: `\usepackage[intlimits]{amsmath}` this way, the integral symbols will be handled like limits and they will look like $\int\limits_a^b$.

If you want to write multiple-line subscripts you have to use `\substack{...}`. This is particularly useful for sums, but you could need it for integrals as well. Just place `\substack` as subscript and put in the argument the output you want. Within the argument of `\substack` you are allowed to use `\\` to start a new line. All the lines you introduce will be centered. Here is an example:

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

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

Math operators

There are operators such as `\sin`, `\cos`, `\log` that LaTeX handles in a special way: it prints them in roman instead of italics and leaves the right space before and after them. There are lot but you might want to make your own. With amsmath it is very easy, just use the following command (it is explained with an example):

`\DeclareMathOperator{\ustep}{ustep}` the first argument is the command you define, the second one is the text it will print. LaTeX will take care of font-formatting and spacing. There is also a starred version `\DeclareMathOperator*{...}{...}`, it works the same but the operator you define will be handled like `\lim`, so the subscripts will be placed under the operator instead of the bottom right like all the others.

Fractions and binomials

Besides the standard `\frac` command to add fractions, amsmath adds two new commands. They use the same syntax but they have a sightly different meaning:

`\dfrac`	forces the fraction to be in display style	equivalent to `\displaystyle \frac`	$\dfrac{x+1}{y^2}$
`\tfrac`	forces the fraction to be in text style	equivalent to `\textstyle \frac`	$\tfrac{x+1}{y^2}$

for example, you could force the display style of a fraction within a matrix environment; using the amsmath shorter versions will keep your source code more readable. For `\binom` the dual commands are defined: `\dbinom` and `\tbinom`.

Text over symbols

If you want to write something over a symbol you can easily do it with the following command: `\overset{top}{symbol}` it will take *top*, resize it and put it over *symbol*. With this command you can easily create new symbols like , that are given by

$$\overset{!}{=} \quad \overset{?}{\leq}$$

`\overset{!}{=}` and `\overset{?}{\leq}`. If you want to write under a symbol, the dual command is `\underset`, that works with the same syntax.

Anyway, if you want to write some text over an arrow, amsmath provides its own command:

`\xleftarrow{up}`	$\xleftarrow{up}$
`\xleftarrow[down]{up}`	$\xleftarrow[down]{up}$

it will create a left-pointing arrow of the right length and will write *up* over it and optionally *down* under it. For right-pointing arrows use `\xrightarrow` with the same syntax.

List of Mathematical Symbols

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

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`	$\doteq$	`\doteq`
$\frown$	`\frown`	$\in$	`\in`	$\ni$	`\ni`
$\propto$	`\propto`	$=$	`=`	$\vdash$	`\vdash`
$\dashv$	`\dashv`	$<$	$<$	$>$	$>$

Table 13.1: Relation Symbols

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`	$\ast$	`\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`		

Table 13.2: Binary Operations

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

Further reading

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

External links

- Latex maths symbols
- `amsmath` documentation