The mathtools package*

Morten Høgholm

2004/10/10

Abstract

The mathtools package is an extension package to amsmath. There are two things on mathtools' agenda: 1) correct various bugs/defeciencies in amsmath until these are fixed by the $\mathcal{AMS}$ and 2) provide useful tools for mathematical typesetting, be it a small macro for typesetting a prescript or an underbraket, or entirely new display math constructs such as a `multlined` environment.

Contents

---

*This file has version number v1.01a, last revised 2004/10/10.

1 Introduction

Although amsmath provides many handy tools for mathematical typesetting, it is nonetheless a static package. This is not a bad thing, because what it does, it mostly does quite well and having a stable math typesetting package is "a good thing." However, amsmath does not fulfill all the needs of the mathematical part of the LaTeX community, resulting in many authors writing small snippets of code for tweaking the mathematical layout. Some of these snippets has also been posted to newsgroups and mailing lists over the years, although more often than not without being released as stand-alone packages.

The mathtools package is exactly what its name implies: tools for mathematical typesetting. It is a collection of many of these often needed small tweaks—with some big tweaks added as well. It can only do so by having me harvesting newsgroups for code and/or you writing me with wishes for code to be included, so if you have any good macros or just macros that help you when writing mathematics, then don't hesitate to report them to me. I can be reached at

    `moho01ab@student.cbs.dk`

This is of course also the address to use in case of bug reports.

2 Package loading

The mathtools package requires amsmath but is able to pass options to it as well. Thus a line like

    `\usepackage[fleqn,tbtags]{mathtools}`

is equivalent to

    `\usepackage[fleqn,tbtags]{amsmath}`
    `\usepackage{mathtools}`

2.1 Special mathtools options

fixamsmath	donotfixamsmathbugs

The option `fixamsmath` (default) fixes two bugs in amsmath.[1] Should you for some reason not want to fix these bugs then just add the option `donotfixamsmathbugs`

---

[1]See the online LaTeX bugs database `http://www.latex-project.org/cgi-bin/ltxbugs2html` under $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX problem reports 3591 and 3614.

(if you can do it without typos). The reason for this extremely long name is that I really don't see why you wouldn't want these bugs to be fixed, so I've made it slightly difficult not to fix them.

```
allowspaces       disallowspaces
```

Sometimes amsmath gives you nasty surprises, as here where things look seemingly innocent:

```
\[
    \begin{gathered}
      [p] = 100 \\
      [v] = 200
    \end{gathered}
\]
```

Without mathtools this will result in this output:

$$= 100$$
$$[v] = 200$$

Yes, the [p] has been gobbled without any warning whatsoever.[2] This is hardly what you'd expect as an end user, as the desired output was probably something like this instead:

$$[p] = 100$$
$$[v] = 200$$

With the option `disallowspaces` (default) mathtools disallows spaces in front of optional arguments where it could possibly cause problems just as amsmath does with \\ inside the display environments. This includes the environments `gathered` (and also those shown in § 4.5 on page 19), `aligned`, `multlined`, and the extended `matrix`-environments (§ 3.4.1 on page 12). If you however want to preserve the more dangerous standard optional spaces, simply choose the option `allowspaces`.

3   Tools for mathematical typesetting

```
\mathtoolsset{⟨key val list⟩}
```

Many of the tools shown in this manual can be turned on and off be setting a switch to either true or false. In all cases it is done with the command \mathtoolsset. A typical use could be something like

```
\mathtoolsset{
  showonlyrefs,
  mathic % or mathic = true
}
```

More information on the keys later on.

---

[2] amsmath thought the [p] was an optional argument, checked if it was t or b and when both tests failed, assumed it was a c.

3.1 Fine-tuning mathematical layout

Sometimes you need to tweak the layout of formulas a little to get the best result and this part of the manual describes the various macros mathtools provides for this.

3.1.1 A complement to \smash, \llap, and \rlap

```
\mathllap[⟨mathstyle⟩]{⟨math⟩}    \mathclap[⟨mathstyle⟩]{⟨math⟩}
\mathrlap[⟨mathstyle⟩]{⟨math⟩}    \clap{⟨text⟩}
\mathmbox{⟨math⟩}                 \mathmakebox[⟨width⟩][⟨pos⟩]{⟨math⟩}
```

In [1], Alexander R. Perlis describes some simple yet useful macros for use in math displays. For example the display

```
\[
  X = \sum_{1\le i\le j\le n} X_{ij}
\]
```

$$X = \sum_{1 \le i \le j \le n} X_{ij}$$

contains a lot of excessive white space. The idea that comes to mind is to fake the width of the subscript. The command \mathclap puts its argument in a zero width box and centers it, so it could possibly be of use here.

```
\[
  X = \sum_{\mathclap{1\le i\le j\le n}} X_{ij}
\]
```

$$X = \sum_{1 \le i \le j \le n} X_{ij}$$

For an in-depth discussion of these macros I find it better to read the article; an online version can be found at

http://www.tug.org/TUGboat/Articles/tb22-4/tb72perlS.pdf

Note that the definitions shown in the article do not exactly match the definitions in mathtools. Besides providing an optional argument for specifying the desired math style, these versions also work around a most unfortunate TeX "feature."[3] The \smash macro is fixed too.

3.1.2 Forcing a cramped style

```
\cramped[⟨mathstyle⟩]{⟨math⟩}
```

Let's look at another example where we have used \mathclap:

```
\begin{equation}\label{eq:mathclap}
  \sum_{\mathclap{a^2<b^2<c}}\qquad
  \sum_{a^2<b^2<c}
\end{equation}
```

---

[3]The faulty reboxing procedure.

$$\sum_{a^2<b^2<c} \quad \sum_{a^2<b^2<c} \tag{1}$$

Do you see the difference? Maybe if I zoomed in a bit:

$$\sum_{a^2<b^2<c} \qquad \sum_{a^2<b^2<c}$$

Notice how the limit of the right summation sign is typeset in a more compact style than the left. It is because TeX sets the limits of operators in a *cramped* style. For each of TeX' four math styles (`\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle`), there also exists a cramped style that doesn't raise exponents as much. Besides in the limits of operators, TeX also automatically uses these cramped styles in radicals such as `\sqrt` and in the denominators of fractions, but unfortunately there are no primitive commands that allows you to detect crampedness or switch to it.

mathtools offers the command `\cramped` which forces a cramped style in normal un-cramped math. Additionally you can choose which of the four styles you want it in as well by specifying it as the optional argument:

```
\[
  \cramped{x^2}              \leftrightarrow x^2    \quad
  \cramped[\scriptstyle]{x^2} \leftrightarrow {\scriptstyle x^2}
\]
```

$$x^2 \leftrightarrow x^2 \quad x^2 \leftrightarrow x^2$$

You may be surprised how often the cramped style can be beneficial yo your output. Take a look at this example:

```
\begin{quote}
  The 2005 Euro\TeX{} conference is held in Abbaye des
  Pr\'emontr\'es, France, marking the 16th ($2^{2^2}$) anniversary
  of both Dante and GUTenberg (the German and French \TeX{} users
  group resp.).
\end{quote}
```

> The 2005 EuroTeX conference is held in Abbaye des Prémontrés, France, marking the 16th (2^{2^2}) anniversary of both Dante and GUTenberg (the German and French TeX users group resp.).

Typesetting on a grid is generally considered quite desirable, but as the second line of the example shows, the exponents of 2 causes the line to be too tall for the normal value of `\baselineskip`, so TeX inserts a `\lineskip` (normal value is 1.0pt). In order to circumvent the problem, we can force a cramped style so that the exponents aren't raised as much:

```
\begin{quote}
  The 2005 Euro\TeX{} ... 16th ($\cramped{2^{2^2}}$) ...
\end{quote}
```

5

The 2005 EuroTEX conference is held in Abbaye des Prémontrés, France, marking the 16th (2^{2^2}) anniversary of both Dante and GUTenberg (the German and French TEX users group resp.).

\crampedllap[⟨*mathstyle*⟩]{⟨*math*⟩}   \crampedclap[⟨*mathstyle*⟩]{⟨*math*⟩}
\crampedrlap[⟨*mathstyle*⟩]{⟨*math*⟩}

The commands \crampedllap, \crampedclap, and \crampedrlap are identical to the three \mathXlap commands described earlier except the argument is typeset in cramped style. You need this in order to typeset (1) correctly while still faking the width of the limit.

```
\begin{equation*}\label{eq:mathclap-b}
  \sum_{\crampedclap{a^2<b^2<c}}
  \tag{\ref{eq:mathclap}*}
\end{equation*}
```

$$\sum_{a^2<b^2<c} \tag{1*}$$

Of course you could just type

```
\sum_{\mathclap{\cramped{a^2<b^2<c}}}
```

but it has one major disadvantage: In order for \mathXlap and \cramped to get the right size, TEX has to process them four times, meaning that nesting them as shown above will cause TEX to typeset 4^2 instances before choosing the right one. In this situation however, we will of course need the same style for both commands so it makes sense to combine the commands in one, thus letting TEX make the choice only once rather than twice.

3.1.3 Smashing an operator

\smashoperator[⟨*pos*⟩]{⟨*operator with limits*⟩}

Above we shoved how to get LATEX to ignore the width of the subscript of an operator. However this approach takes a lot of extra typing, especially if you have a wide superscript, meaning you have to put in \crampedclap in both sub- and superscript. To make things easier, mathtools provides a \smashoperator command, which simply ignores the width of the sub- and superscript. It also takes an optional argument, l, r, or lr (default), denoting which side of the operator should be ignored (smashed).

```
\[
  V = \sum_{1\le i\le j\le n}^{\infty} V_{ij}                 \quad
  X = \smashoperator{\sum_{1\le i\le j\le n}^{3456}} X_{ij}    \quad
  Y = \smashoperator[r]{\sum\limits_{1\le i\le j\le n}} Y_{ij} \quad
  Z = \smashoperator[l]{\mathop{T}_{1\le i\le j\le n}} Z_{ij}
\]
```

$$V = \sum_{1\le i\le j\le n}^{\infty} V_{ij} \quad X = \sum_{1\le i\le j\le n}^{3456} X_{ij} \quad Y = \sum_{1\le i\le j\le n} Y_{ij} \quad Z = T_{1\le i\le j\le n} Z_{ij}$$

Note that \smashoperator always sets its argument in display style and with limits even if you have used the nosumlimits option of amsmath. If you wish, you can use shorthands for _ and ^ such as \sb and \sp.

6

3.1.4 Adjusting limits of operators

```
\adjustlimits{⟨operator₁⟩}_{⟨limit₁⟩} {⟨operator₂⟩}_{⟨limit₂⟩}
```

When typesetting two consecutive operators with limits one often wishes the limits of the operators were better aligned. Look closely at these examples:

```
\[
  \text{a)} \lim_{n\to\infty} \max_{p\ge n} \quad
  \text{b)} \lim_{n\to\infty} \max_{p^2\ge n} \quad
  \text{c)} \lim_{n\to\infty} \sup_{p^2\ge nK} \quad
  \text{d)} \limsup_{n\to\infty} \max_{p\ge n}
\]
```

$$\text{a)} \lim_{n\to\infty} \max_{p\ge n} \quad \text{b)} \lim_{n\to\infty} \max_{p^2\ge n} \quad \text{c)} \lim_{n\to\infty} \sup_{p^2\ge nK} \quad \text{d)} \limsup_{n\to\infty} \max_{p\ge n}$$

a) looks okay, but b) is not quite as good because the second limit ($p^2 \ge n$) is significantly taller than the first ($n \to \infty$). With c) things begin to look really bad, because the second operator has a descender while the first doesn't, and finally we have d) which looks just as bad as c). The command \adjustlimits is useful in these cases, as you can just put it in front of these consecutive operators and it'll make the limits line up.

```
\[
  \text{a)} \adjustlimits\lim_{n\to\infty} \max_{p\ge n} \quad
  \text{b)} \adjustlimits\lim_{n\to\infty} \max_{p^2\ge n} \quad
  \text{c)} \adjustlimits\lim_{n\to\infty} \sup_{p^2\ge nK} \quad
  \text{d)} \adjustlimits\limsup_{n\to\infty} \max_{p\ge n}
\]
```

$$\text{a)} \lim_{n\to\infty} \max_{p\ge n} \quad \text{b)} \lim_{n\to\infty} \max_{p^2\ge n} \quad \text{c)} \lim_{n\to\infty} \sup_{p^2\ge nK} \quad \text{d)} \limsup_{n\to\infty} \max_{p\ge n}$$

The use of \sb instead of _ is allowed.

3.2 Controlling tags

In this section various tools for altering the appearance of tags are shown. All of the tools here can be used at any point in the document but they should probably be affect the whole document, so the preamble is the best place to issue them.

3.2.1 The appearance of tags

```
\newtagform{⟨name⟩}[⟨inner_format⟩]{⟨left⟩}{⟨right⟩}
\renewtagform{⟨name⟩}[⟨inner_format⟩]{⟨left⟩}{⟨right⟩}
\usetagform{⟨name⟩}
```

Altering the layout of equation numbers in amsmath is not very user friendly (it involves a macro with three @'s in its name), so mathtools provides an interface somewhat reminiscent of the page style concept. This way you can define several different tag forms and then choose the one you prefer.

As an example let's try to define a tag form which puts the equation number in square brackets. First we define a brand new tag form:

```
\newtagform{brackets}{[}{]}
```

Then we activate it:

```
\usetagform{brackets}
```

The result is then

$$E \neq mc^3 \tag*{[2]}$$

Similarly you could define a second version of the brackets that prints the equation number in bold face instead

```
\newtagform{brackets2}[\textbf]{[}{]}
\usetagform{brackets2}
\begin{equation}
  E \neq m c^3
\end{equation}
```

$$E \neq mc^3 \tag*{[\mathbf{3}]}$$

When you reference an equation with \eqref, the tag form in effect at the time of referencing controls the formatting, so be careful if you use different tag forms throughout your document.

If you want to renew a tag form, then use the command \renewtagform. Should you want to return to the standard setting then choose

```
\usetagform{default}
```

3.2.2 Showing only referenced tags

> showonlyrefs = true|false
> showmanualtags = true|false
> \refeq{⟨*label*⟩}

An equation where the tag is produced with a manual \tag* shouldn't be referenced with the normal \eqref because that would format it according to the current tag format. Using just \ref on the other hand may not be a good solution either as the argument of \tag* is always set in upright shape in the equation and you may be referencing it in italic text. In the example below, the command \refeq is used to avoid what could possibly lead to confusion in cases where the tag font has very different form in upright and italic shape (here we switch to Palatino in the example):

```
\begin{quote}\renewcommand*\rmdefault{ppl}\normalfont\itshape
\begin{equation*}
  a=b \label{eq:example}\tag*{Q\&A}
\end{equation*}
See \ref{eq:example} or is it better with \refeq{eq:example}?
\end{quote}
```

$$a = b \qquad \text{Q\&A}$$

See Q&A or is it better with Q&A?

Another problem sometimes faced is the need for showing the equation numbers for only those equations actually referenced. In mathtools this can be done by setting the key showonlyrefs to either true or false by using `\mathtoolsset`. You can also choose whether or not to show the manual tags specified with `\tag` or `\tag*` by setting the option showmanualtags to true or false.[4] For both keys just typing the name of it chooses true as shown in the following example.

```
\mathtoolsset{showonlyrefs,showmanualtags}
\usetagform{brackets}
\begin{gather}
  a=a \label{eq:a} \\
  b=b \label{eq:b} \tag{**}
\end{gather}
This should refer to the equation containing $a=a$: \eqref{eq:a}.
Then a switch of tag forms.
\usetagform{default}
\begin{align}
  c&=c \label{eq:c} \\
  d&=d \label{eq:d}
\end{align}
This should refer to the equation containing $d=d$: \eqref{eq:d}.
\begin{equation}
  e=e
\end{equation}
Back to normal.\mathtoolsset{showonlyrefs=false}
\begin{equation}
  f=f
\end{equation}
```

$$a = a \qquad [4]$$
$$b = b \qquad [**]$$

This should refer to the equation containing $a = a$: [4]. Then a switch of tag forms.

$$c = c$$
$$d = d \qquad (5)$$

This should refer to the equation containing $d = d$: (5).

$$e = e$$

Back to normal.

$$f = f \qquad (6)$$

Note that this feature only works if you use `\eqref` or `\refeq` to reference your equations.

---

[4]I recommend setting showmanualtags to true, else the whole idea of using `\tag` doesn't really make sense, does it?

3.3 Extensible symbols

The number of horizontally extensible symbols in standard LaTeX and amsmath is somewhat low. This part of the manual describes what mathtools does to help this situation.

3.3.1 Arrow-like symbols

```
\xleftrightarrow[⟨sub⟩]{⟨sup⟩}          \xRightarrow[⟨sub⟩]{⟨sup⟩}
\xLeftarrow[⟨sub⟩]{⟨sup⟩}               \xLeftrightarrow[⟨sub⟩]{⟨sup⟩}
\xhookleftarrow[⟨sub⟩]{⟨sup⟩}           \xhookrightarrow[⟨sub⟩]{⟨sup⟩}
\xmapsto[⟨sub⟩]{⟨sup⟩}
```

Extensible arrows are part of amsmath in the form of the commands

\xrightarrow[⟨*subscript*⟩]{⟨*superscript*⟩}   and
\xleftarrow[⟨*subscript*⟩]{⟨*superscript*⟩}

But what about extensible versions of say, \leftrightarrow or \Longleftarrow? It turns out that the above mentioned extensible arrows are the only two of their kind defined by amsmath, but luckily mathtools helps with that. The extensible arrow-like symbols in mathtools follow the same naming scheme as the ones in amsmath so to get an extensible \Leftarrow you simply do a

```
\[
  A \xLeftarrow[under]{over} B
\]
```

$$A \xLeftarrow[under]{over} B$$

```
\xrightharpoondown[⟨sub⟩]{⟨sup⟩}        \xrightharpoonup[⟨sub⟩]{⟨sup⟩}
\xleftharpoondown[⟨sub⟩]{⟨sup⟩}         \xleftharpoonup[⟨sub⟩]{⟨sup⟩}
\xrightleftharpoons[⟨sub⟩]{⟨sup⟩}       \xleftrightharpoons[⟨sub⟩]{⟨sup⟩}
```

mathtools also provides the extensible harpoons shown above. They're taken from [6].

3.3.2 Braces and brackets

LaTeX defines other kinds of extensible symbols like \overbrace and \underbrace, but sometimes you may want another symbol, say, a bracket.

```
\underbracket[⟨rule thickness⟩] [⟨bracket height⟩]{⟨arg⟩}
\overbracket[⟨rule thickness⟩] [⟨bracket height⟩]{⟨arg⟩}
```

The commands \underbracket and \overbracket are inspired by [6], although the implementation here is slightly different. Used without the optional arguments the bracket commands produce this:

$\underbracket {foo\ bar}_{baz}$ $\underbrace{foo\ bar}_{baz}$

$\overbracket {foo\ bar}^{baz}$ $\overbrace{foo\ bar}^{baz}$

The default rule thickness is equal to that of \underbrace (app. 5/18 ex) while the default bracket height is equal to app. 0.7 ex. These values give really pleasing results in all font sizes, but feel free to use the optional arguments. That way you may get "beauties" like

```
\[
  \underbracket[3pt]{xxx\  yyy}_{zzz} \quad \text{and} \quad
  \underbracket[1pt][7pt]{xxx\  yyy}_{zzz}
\]
```

$$\underbracket{xxx\ yyy}_{zzz} \quad \text{and} \quad \underbracket{xxx\ yyy}_{zzz}$$
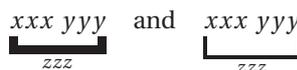
\underbrace{⟨*arg*⟩}	\LaTeXunderbrace{⟨*arg*⟩}
\overbrace{⟨*arg*⟩}	\LaTeXoverbrace{⟨*arg*⟩}

The standard implementation of the math operators \underbrace and \overbrace in LaTeX has some deficiencies. For example, all lengths used internally are *fixed* and optimized for 10 pt typesetting. As a direct consequence thereof, using font sizes other than 10 will produce less than optimal results. Another unfortunate feature is the size of the braces. In the example below, notice how the math operator \sum places its limit compared to \underbrace.

$$\sum_n \underbrace{foof}_{zzz}$$
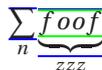
The blue lines indicate the dimensions of the math operator and the green lines the dimensions of $foof$. As you can see, there seems to be too much space between the brace and the zzz whereas the space between brace and $foof$ is okay. Let's see what happens when we use a bigger font size:

$$\sum_n \underbrace{foof}_{zzz}$$

Now there's to little space between the brace and the zzz and also too little space between the brace and the $foof$. If you use Computer Modern you'll actually see that the f overlaps with the brace! Let's try in \footnotesize:

$$\sum_n \underbrace{foof}_{zzz}$$

Here the spacing above and below the brace is quite excessive.

As \overbrace has the exact same problems, there are good reasons for math-tools to make redefinitions of \underbrace and \overbrace. These new versions work equally well in all font sizes and fixes the spacing issues and apart from working with the default Computer Modern fonts, they also work with the packages mathpazo, pamath, fourier, eulervm, cmbright, and mathptmx. If you use the ccfonts to get the full Concrete fonts, the original version saved under the names \LaTeXunderbrace and \LaTeXoverbrace are better, due to of the special design of the Concrete extensible braces. In that case you should probably just add the lines

```
\let\underbrace\LaTeXunderbrace
\let\overbrace\LaTeXoverbrace
```

to your preamble after loading mathtools which will restore the original definitions of \overbrace and \underbrace.

3.4   New mathematical building blocks

In this part of the manual, various mathematical environments are described.

3.4.1   Matrices

```
\begin{matrix*} [⟨col⟩] ⟨contents⟩ \end{matrix*}
\begin{pmatrix*}[⟨col⟩] ⟨contents⟩ \end{pmatrix*}
\begin{bmatrix*}[⟨col⟩] ⟨contents⟩ \end{bmatrix*}
\begin{Bmatrix*}[⟨col⟩] ⟨contents⟩ \end{Bmatrix*}
\begin{vmatrix*}[⟨col⟩] ⟨contents⟩ \end{vmatrix*}
\begin{Vmatrix*}[⟨col⟩] ⟨contents⟩ \end{Vmatrix*}
```

All of the amsmath `matrix` environments center the columns by default, which is not always what you want. Thus mathtools provides a starred version for each of the original environments. These starred environments take an optional argument specifying the alignment of the columns, so that

```
\[
  \begin{pmatrix*}[r]
    -1 & 3 \\
     2  & -4
  \end{pmatrix*}
\]
```

yields

$$\begin{pmatrix} -1 & 3 \\ 2 & -4 \end{pmatrix}$$

The optional argument (default is `[c]`) can be any column type valid in the usual `array` environment.

3.4.2   The `multlined` environment

```
\begin{multlined}[⟨pos⟩][⟨width⟩] ⟨contents⟩ \end{multlined}
\shoveleft[⟨dimen⟩]{⟨arg⟩}    \shoveright[⟨dimen⟩]{⟨arg⟩}
firstline-afterskip = ⟨dimen⟩      lastline-preskip = ⟨dimen⟩
multlined-width = ⟨dimen⟩          multlined-pos = c|b|t
```

Some of the amsmath environments exist in two forms: an outer and an inner environment. One example is the pair `gather` & `gathered`. There is one important omission on this list however, as there is no inner `multlined` environment, so this is where mathtools steps in.

One might wonder what the sensible behavior should be. We want it to be an inner environment so that it is not wider than necessary, but on the other hand we would like to be able to control the width. The current implementation of `multlined` handles both cases. The idea is this: Set the first line flush left and add a hard space after it; this space is governed by the firstline-afterskip key. The last line should be

set flush right and preceded by a hard space of size lastline-preskip. Both these hard spaces have a default value of \multlinegap. Here we use a 't' in the first optional argument denoting a top-aligned building block (the default is 'c').

```
\[
  A = \begin{multlined}[t]
        \framebox[4cm]{first} \\
        \framebox[4cm]{last}
      \end{multlined} B
\]
```

$$A = \boxed{\qquad\qquad \text{first} \qquad\qquad} \quad B$$
$$\boxed{\qquad\qquad \text{last} \qquad\qquad}$$
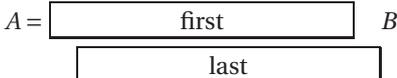
Note also that multlined gives you access to an extended syntax for \shoveleft and \shoveright as shown in the example below.

```
\[
  \begin{multlined}
    \framebox[.65\columnwidth]{First line}          \\
    \framebox[.5\columnwidth]{Second line}          \\
    \shoveleft{L+E+F+T}                             \\
    \shoveright{R+I+G+H+T}                          \\
    \shoveleft[1cm]{L+E+F+T}                        \\
    \shoveright[\widthof{$R+I+G+H+T$}]{R+I+G+H+T} \\
    \framebox[.65\columnwidth]{Last line}
  \end{multlined}
\]
```

$$\boxed{\qquad\qquad\quad \text{First line} \qquad\qquad\quad}$$
$$\boxed{\qquad\quad \text{Second line} \qquad\quad}$$
$$L + E + F + T$$
$$R + I + G + H + T$$
$$L + E + F + T$$
$$R + I + G + H + T$$
$$\boxed{\qquad\qquad\quad \text{Last line} \qquad\qquad\quad}$$
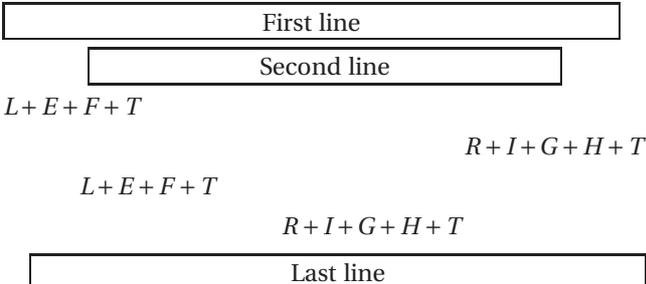
You can also choose the width yourself by specifying it as an optional argument:

```
\[
  \begin{multlined}[b][7cm]
    \framebox[4cm]{first} \\
    \framebox[4cm]{last}
  \end{multlined} = B
\]
```

$$\boxed{\qquad\quad \text{first} \qquad\quad}$$
$$\boxed{\qquad\quad \text{last} \qquad\quad} = B$$
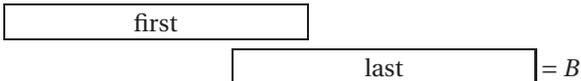
There can be two optional arguments (position and width) and they're interchangeable.

3.4.3 More cases-like environments

```
\begin{dcases}  ⟨math_column⟩ & ⟨math_column⟩ \end{dcases}
\begin{dcases*} ⟨math_column⟩ & ⟨text_column⟩   \end{dcases*}
```

Anyone who have tried to use an integral in the regular `cases` environment from amsmath will have noticed that it is set as

$$a = \begin{cases} E = mc^2 & \text{Nothing to see here} \\ \int x - 3\, dx & \text{Integral is text style} \end{cases}$$

mathtools provides two environments similar to `cases`. Using the `dcases` environment you get the same output as with `cases` except that the rows are set in display style.

```
\[
  \begin{dcases}
    E = m c^2       & c \approx 3.00\times 10^{8}\,\mathrm{m}/\mathrm{s} \\
    \int x-3\, dx & \text{Integral is display style}
  \end{dcases}
\]
```

$$\begin{dcases} E = mc^2 & c \approx 3.00 \times 10^8 \, \mathrm{m/s} \\ \int x - 3\, dx & \text{Integral is display style} \end{dcases}$$

Additionally the environment `dcases*` acts just the same, but the second column is set in the normal roman font of the document.[5]

```
\[
  a= \begin{dcases*}
    E = m c^2       & Nothing to see here \\
    \int x-3\, dx & Integral is display style
  \end{dcases*}
\]
```

$$a = \begin{dcases*} E = mc^2 & \text{Nothing to see here} \\ \int x - 3\, dx & \text{Integral is display style} \end{dcases*}$$

3.5 Short intertext

```
\shortintertext{⟨text⟩}
```

amsmath provides the command `\intertext` for interrupting a multi line display while still maintaining the alignment points. However the spacing often seems quite excessive as seen below.

```
\begin{align}
  a&=b \intertext{Some text}
  c&=d
\end{align}
```

---

[5]Or rather: it inherits the font characteristics active just before the `dcases*` environment.

$$a = b \tag{7}$$

Some text

$$c = d \tag{8}$$

Using the command \shortintertext alleviates this situation somewhat:

```
\begin{align}
  a&=b \shortintertext{Some text}
  c&=d
\end{align}
```

$$a = b \tag{9}$$

Some text

$$c = d \tag{10}$$

3.6 Paired delimiters

Feature request by
Lars Madsen
2004/06/25

> \DeclarePairedDelimiter{⟨*cmd*⟩}{⟨*left_delim*⟩}{⟨*right_delim*⟩}

In the amsmath documentation it is shown how to define a few commands for type-setting the absolute value and norm. These definitions are:

```
\newcommand*\abs[1]{\lvert#1\rvert}
\newcommand*\norm[1]{\lVert#1\rVert}
```

While they produce correct horizontal spacing you have to be careful about the vertical spacing if the argument is just a little taller than usual as in

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

Here it won't give a nice result, so you have to manually put in either \left–\right pair or a \bigl–\bigr pair. Both methods mean that you have to delete your \abs command, which may not sound like an ideal solution.

With the command \DeclarePairedDelimiter you can combine all these features in one easy to use command. Let's show an example:

```
\DeclarePairedDelimiter\abs{\lvert}{\rvert}
```

This defines the command \abs just like in the amsmath documentation but with a few additions:

- A starred variant: \abs* produces delimiters that are preceded by \left and \right resp.:

  ```
  \[
    \abs*{\frac{a}{b}}
  \]
  ```

$$\left|\frac{a}{b}\right|$$

15

- A variant with an optional argument: \abs[⟨*size_cmd*⟩], where ⟨*size_cmd*⟩ is either \big, \Big, \bigg, or \Bigg (if you have any bigggger versions you can use them too).

```
\[
  \abs[\Bigg]{\frac{a}{b}}
\]
```

$$\left|\frac{a}{b}\right|$$

3.7 Special symbols

This part of the manual is about special symbols. So far only one technique is covered, but more will come.

3.7.1 Left and right parentheses

\lparen   \rparen

When you want a big parenthesis or bracket in a math display you usually just type

```
\left( ... \right)  or  \left[ ... \right]
```

LaTeX also defines the macro names \lbrack and \rbrack to be shorthands for the left and right square bracket resp., but doesn't provide similar definitions for the parentheses. Some packages need command names to work with[6] so mathtools defines the commands \lparen and \rparen to represent the left and right parenthesis resp.

3.7.2 Vertically centered colon

centercolon = true|false
\vcentcolon   \ordinarycolon

When trying to show assignment operations as in $a := b$, one quickly notices that the colon is not centered on the math axis as the equal sign, leading to an odd-looking output. The command \vcentcolon is a shorthand for such a vertically centered colon, and can be used as in $a \vcentcolon= b$ and results in the desired output: $a := b$.

Typing \vcentcolon every time is quite tedious, so one can use the key center-colon to make the colon active instead.

```
\mathtoolsset{centercolon}
\[
  a := b
\]
\mathtoolsset{centercolon=false}
```

---

[6]The empheq package needs command names for delimiters in order to make auto-scaling versions.

16

$$a := b$$

In this case the command \ordinarycolon typesets an ... ordinary colon (what a surprise).

\coloneqq	\Coloneqq	\coloneq	\Coloneqq
\eqqcolon	\Eqqcolon	\eqcolon	\Eqcolon
\colonapprox	\Colonapprox	\colonsim	\Colonsim
\dblcolon			

The font packages txfonts and pxfonts provides various symbols that include a vertically centered colon but with tighter spacing. For example, the combination := exists as the symbol \coloneqq which typesets as := instead of :=. The primary disadvantage of using these fonts are the support packages' lack of support for amsmath (and thus mathtools) and worse yet, the side-bearings are way too tight; see [4] for examples. If you're not using these fonts, mathtools provides the symbols for you. Here are a few examples:

```
\[
  a \coloneqq b \quad c \Colonapprox d \quad e \dblcolon f
\]
```

$$a := b \quad c :\!\approx d \quad e :: f$$

4 A tribute to Michael J. Downes

Michael J. Downes (1958–2003) was one of the major architects behind amsmath and member of the LATEX Team. He made many great contributions to the TEX community; not only by the means of widely spread macro packages such as amsmath but also in the form of actively giving advice on newsgroups. Some of Michael's macro solutions on the newsgroups never made it into publicly available macro packages although they certainly deserved it, so mathtools tries to rectify this matter. The macros described in this section are either straight copies or heavily inspired by his original postings.

4.1 Mathematics within italic text

Posted on `comp.text.tex`
Michael J. Downes
1998/05/14

mathic = true|false

TEX usually takes care of italic corrections in text, but fails when it comes to math. If you use the LATEX inline math commands \(and \) you can however work around it by setting the key mathic to true as shown below.

```
\begin{quote}\itshape
Compare these lines: \par
\mathtoolsset{mathic} % or \mathtoolsset{mathic=true}
Subset of \(V\) and subset of \(A\). \par
\mathtoolsset{mathic=false}
Subset of \(V\) and subset of \(A\).
\par
\end{quote}
```

Compare these lines:

Subset of V and subset of A.

Subset of V and subset of A.

4.2 Left sub/superscripts

| `\prescript{`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*arg*⟩`}`	prescript-sup-format = ⟨*cmd*⟩
prescript-sub-format = ⟨*cmd*⟩	prescript-arg-format = ⟨*cmd*⟩

Sometimes one wants to put a sub- or superscript on the left of the argument. The `\prescript` command does just that:

```
\[
  {}^{4}_{12}\mathbf{C}^{5+}_{2}          \quad
  \prescript{14}{2}{\mathbf{C}}^{5+}_{2}  \quad
  \prescript{4}{12}{\mathbf{C}}^{5+}_{2}  \quad
  \prescript{14}{}{\mathbf{C}}^{5+}_{2}   \quad
  \prescript{}{2}{\mathbf{C}}^{5+}_{2}
\]
```

$$\prescript{4}{12}{\mathbf{C}}^{5+}_{2} \quad \prescript{14}{2}{\mathbf{C}}^{5+}_{2} \quad \prescript{4}{12}{\mathbf{C}}^{5+}_{2} \quad \prescript{14}{}{\mathbf{C}}^{5+}_{2} \quad \prescript{}{2}{\mathbf{C}}^{5+}_{2}$$

The formatting of the arguments is controlled by three keys. This silly example shows you how to use them:

```
\newcommand*\myisotope[3]{%
  \begingroup % to keep changes local. We cannot use a brace group
              % as it affects spacing!
  \mathtoolsset{
    prescript-sup-format=\mathit,
    prescript-sub-format=\mathbf,
    prescript-arg-format=\mathrm,
  }%
  \prescript{#1}{#2}{#3}%
  \endgroup
}
\[
  \myisotope{A}{Z}{X}\to \myisotope{A-4}{Z-2}{Y}+
  \myisotope{4}{2}{\alpha}
\]
```

$$\prescript{A}{Z}{\mathrm{X}} \to \prescript{A-4}{Z-2}{\mathrm{Y}} + \prescript{4}{2}{\alpha}$$

4.3 Declaring math sizes

`\DeclareMathSizes{`⟨*dimen*⟩`}{`⟨*dimen*⟩`}{`⟨*dimen*⟩`}{`⟨*dimen*⟩`}`

If you don't know about `\DeclareMathSizes`, then skip the rest of this text. If you do know, then all that is needed to say is that with mathtools it is patched so that all regular dimension suffixes are now valid in the last three arguments. Thus a declaration such as

```
\DeclareMathSize{9.5dd}{9.5dd}{7.5dd}{6.5dd}
```

will now work (it doesn't in standard LaTeX). When this bug has been fixed in LaTeX, this fix will be removed from mathtools.

4.4 Spreading equations

> \begin{spreadlines}{⟨*dimen*⟩} ⟨*contents*⟩ \end{spreadlines}

The spacing between lines in a multi line math environment such as gather is governed by the dimension \jot. The spreadlines environment takes one argument denoting the value of \jot inside the environment:

```
\begin{spreadlines}{20pt}
Large spaces between the lines.
\begin{gather}
  a=b\\
  c=d
\end{gather}
\end{spreadlines}
Back to normal spacing.
\begin{gather}
  a=b\\
  c=d
\end{gather}
```

Large spaces between the lines.

$$a = b \tag{11}$$

$$c = d \tag{12}$$

Back to normal spacing.

$$a = b \tag{13}$$
$$c = d \tag{14}$$

4.5 Gathered environments

> \begin{lgathered}[⟨*pos*⟩] ⟨*contents*⟩ \end{lgathered}
> \begin{rgathered}[⟨*pos*⟩] ⟨*contents*⟩ \end{rgathered}
> \newgathered{⟨*name*⟩}{⟨*pre_line*⟩}{⟨*post_line*⟩}{⟨*after*⟩}
> \renewgathered{⟨*name*⟩}{⟨*pre_line*⟩}{⟨*post_line*⟩}{⟨*after*⟩}

In a document set in fleqn, you might sometimes want an inner gathered environment that doesn't center its lines but puts them flush left. The lgathered environment works just like the standard gathered except that it flushes its contents left:

```
\begin{equation}
  \begin{lgathered}
    x=1,\quad x+1=2 \\
```

```
        y=2
      \end{lgathered}
    \end{equation}
```

$$x = 1, \quad x + 1 = 2$$
$$y = 2 \tag{15}$$

Similarly the `rgathered` puts it contents flush right.

More interesting is probably the command `\newgathered`. In this example we define a gathered version that centers the lines and also prints a star and a number at the left of each line.

```
\newcounter{steplinecnt}
\newcommand\stepline{\stepcounter{steplinecnt}\thesteplinecnt}
\newgathered{stargathered}
          {\llap{\stepline}$*$\quad\hfil}%  \hfil for centering
          {\hfil}%                          \hfil for centering
          {\setcounter{steplinecnt}{0}}%    reset counter
```

With these definitions we can get something like this:

```
\begin{gather}
  \begin{stargathered}
    x=1,\quad x+1=2 \\
    y=2
  \end{stargathered}
\end{gather}
```

$$1* \quad x = 1, \quad x + 1 = 2$$
$$2* \qquad y = 2 \tag{16}$$

`\renewgathered` renews a gathered environment of course.

In all fairness it should be stated that the original concept by Michael has been extended quite a bit in mathtools. Only the end product of `lgathered` is the same.

4.6 Split fractions

$$\boxed{\texttt{\textbackslash splitfrac\{}\langle\textit{numer}\rangle\texttt{\}\{}\langle\textit{denom}\rangle\texttt{\}} \quad \texttt{\textbackslash splitdfrac\{}\langle\textit{numer}\rangle\texttt{\}\{}\langle\textit{denom}\rangle\texttt{\}}}$$

These commands provide split fractions e.g., multi line fractions:

```
\[
  a=\frac{
      \splitfrac{xy + xy + xy + xy + xy}
               {+ xy + xy + xy + xy}
    }
    {z}
  =\frac{
      \splitdfrac{xy + xy + xy + xy + xy}
               {+ xy + xy + xy + xy}
    }
    {z}
\]
```

$$a = \frac{\begin{array}{c}xy + xy + xy + xy + xy\\ + xy + xy + xy + xy\end{array}}{z} = \frac{\begin{array}{c}xy + xy + xy + xy + xy\\ + xy + xy + xy + xy\end{array}}{z}$$

20

References

[1] Alexander R. Perlis, *A complement to \smash, \llap, and \rlap*, TUGboat 22(4) (2001).

[2] American Mathematical Society and Michael Downes, *Technical notes on the amsmath package* Version 2.0, 1999/10/29. (Available from CTAN as file `technote.tex`.)

[3] Frank Mittelbach, Rainer Schöpf, Michael Downes, and David M. Jones, *The amsmath package* Version 2.13, 2000/07/18. (Available from CTAN as file `amsmath.dtx`.)

[4] Frank Mittelbach and Michel Goossens. *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, Massachusetts, 2 edition, 2004. With Johannes Braams, David Carlisle, and Chris Rowley.

[5] David Carlisle, *The keyval Package*, Version 1.13, 1999/03/16. (Available from CTAN as file `keyval.dtx`.)

[6] Herbert Voß, *Math mode*, Version 1.71, 2004/07/06. (Available from CTAN as file `Voss-Mathmode.pdf`.)

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.