Experiences typesetting mathematical physics

Abstract
Twenty years ago, the author was just about to start his university studies in math and physics. A year or so later, he not only discovered a fascinating program called TEX, but he also got involved in a project of typesetting a series of lecture notes which eventually became book manuscripts for a complete course in theoretical physics. In the end, he spent about seven years working on typing, editing, revising, and formatting more than 2500 book pages containing a large amount of math.
While there are many experiences from such a project one could talk about, ranging from issues of project management to document design and layout, this talk will focus on two specific topics: adapting LATEX to deal with the specific requirements of mathematical notation in physics and fine-tuning the appearance of math formulas.

Keywords
math typesetting, physics, notation

Given the conference motto of educational uses of TEX, this paper is based on the author's personal experiences of typesetting a series of lecture notes which eventually became a series of textbooks for a complete course in theoretical physics [1, 2, 3, 4, 5, 6].

Introduction: How I got started typesetting physics

When I started my university studies in math and physics in 1989, I did not know anything about TEX or about typesetting in general. However, I quickly noticed that there was a vast difference in quality of typeset material when it came to math formulas. As it turned out, TEX was already being used by several of the math and physics departments for routine typesetting tasks (such as the weekly handouts of homework exercises to students), while other departments were still using a mathematical typewriter.

I first got to know TEX in the summer of 1990 through some of my friends, who had gotten a TEX distribution from the staff of the university computer center. Unfortunately, I did not have a decent computer to run it until a few months later, so instead of jumping into using TEX right away, I started by reading several books about it, including *The TEXbook* itself. When I eventually got started some time later that year, I soon began looking under the hood, trying to understand how the various bits and pieces of a TEX distribution fit together.

I got started with typesetting physics in early 1991, when the professor who held the theoretical physics course asked for volunteers to help him typeset his lecture notes, so they could be printed and handed out to students. After working on this project for several months on a voluntary basis, I was eventually employed as a student assistant from 1991 to 1995, getting paid to work on editing and typesetting lecture notes with TEX. After finishing my diploma and staying on to work on a PhD project at the same department, I continued to work on book manuscripts until the end of 1998.

From 1991 to 1993 the first editions were prepared for the principal courses in

theoretical physics, consisting of the volumes on mechanics, electrodynamics, quantum mechanics, thermodynamics, and quantum field theory.

From 1994 to 1996 some additional volumes were added for special courses in general relativity, cosmology, and elementary particle theory.

Over time, as I learned more about typography and about typesetting rules for math, I revised the layout and the macros several times and began to write my own document classes and macro packages, switching from LaTeX 2.09 in 1991 to NFSS2 in 1993 and to LaTeX 2_ε in 1994.

Using a full-page layout on A4 paper and Computer Modern fonts at 11 pt type size, the series amounted to 2000 printed pages in total, consisting of six volumes of 300 to 350 pages each and two volumes of 100 pages each.

From 1996 to 1999 the second editions of several volumes were prepared, when the courses were held again for the next generation of students. By this time, a publisher had been found who was interested in publishing the lecture notes as textbooks, but since they already had another series of theoretical physics lectures in their program, it was decided to run several volumes together as an omnibus edition of two large volumes of 1200 to 1300 pages each.

For the publisher edition, the layout was revised using Times and MathTime fonts at 10 pt type size on a smaller page size and using a more compact layout, arriving at about 2500 book pages in total. At the same time the macros for math typesetting were revised once more, taking advantage of some of the extra fonts in the MathTime Plus distribution such as an upright Greek font.

The first volume of the omnibus edition finally appeared in 1999 [1], shortly after I had left university, followed by the second volume in 2004 [2], shortly after my professor retired. (In the meantime, second editions of the remaining volumes were prepared from 1999 to 2003, when the courses were held again.)

After the series was finally completed and deemed successful by the publisher, individual volumes [3, 4, 5, 6] were also published separately in recent years.

In summary, working on this project for seven years from 1991 to 1998 was an interesting experience in a wide range of topics, ranging from project organization to support and maintenance of TeX installations, and from high-level document design of layout to low-level details of math typesetting.

Regarding the technical progress, there are some interesting stories to be told:

In 1991, running a complete book of 300 to 350 pages on a 16 MHz 386 PC required taking a lengthy break and occasionally resulted in crashing TeX, if you forgot to use a BigTeX and had too many labels and references. Usually, running chapters separately with `\includeonly` was the preferred way of working, but this still took several minutes per chapter for each run.

In 1998, running a combined edition of 1200 pages on a 100 MHz 486 PC was already much quicker, but it also required enlarging some parameters in `texmf.cnf` to make it work without running out of memory.

Nowadays, we have GHz computers with GBytes of memory, and modern TeX distributions have become big enough by default, so speed and size are no longer an issue, although it still takes time to process 1200 pages.

On the other hand, getting the fine points of math typesetting right is still far from trivial, so we will concentrate on these topics in the following sections. In the next section of this paper, we will look at the difficulties of how to set up (LA)TeX for properly typesetting physics according to the standards of the field. In the final section, we will look at some examples of how to improve the appearance of math formulas and how to deal with special requirements of notation.

Adapting (LA)TEX for typesetting physics

While TEX, in general, does a very good job of typesetting math, different fields of sciences have slightly different conventions how math should be typeset, and TEX does not support all of them equally well.

By default, TEX's built-in rules for typesetting math-mode material are geared towards the conventions applicable for an American style of typesetting math, as specified by style manuals such as *The Chicago Manual of Style* [7] or the style guides of well-respected math publishers [8]. However, this is only one particular style and not everyone favors the same style. For example, the French tradition may suggest a different style and the Russian tradition yet another one.

In physics and related fields, the established conventions are specified by the handbooks of professional societies, such as the IUPAP red book [9, 10] in physics or the IUPAC green book [11, 12, 13] in physical chemistry, or by international standards such as ISO 31-11 or ISO 80000-2 [14, 15].

In essence, the most important points can be summarized as follows:

- ☐ Symbols for physical quantities should be typeset in math italic.
- ☐ Symbols for vectors should be typeset in bold math italic.
- ☐ Symbols for tensors should be typeset in bold sans serif italic.
- ☐ Symbols for physical units should be typeset in upright roman.
- ☐ Symbols for chemical elements should be typeset in upright roman.
- ☐ Symbols for elementary particles should be typeset in upright roman.
- ☐ Mathematical constants (such as e, i, π) should be upright roman.
- ☐ Mathematical operators (such as d, ∂, δ, Δ) should be upright roman.

In theory, these rules should apply universally to typesetting physics and they should apply to all symbols without exceptions, regardless of whether they are Latin or Greek, uppercase or lowercase. In practice, however, many science publishers insist on their own style guides, which follow the official standards only to a greater or lesser degree [16, 17].

For example, upright bold might be used for vectors instead of bold italic and upright bold sans serif might be used for tensors instead of bold sans serif italic. In addition, the rules for typesetting mathematical constants and mathematical operators are often neglected.

While it is easy to summarize the rules in a few sentences, it is far from easy to implement them using a standard TEX system with a standard set of Computer Modern fonts. In fact, even some recent editions of the guidebooks typeset with TEX suffer from these limitations, so they have become less reliable than some of the earlier editions typeset with traditional methods.

Given the default setup of (LA)TEX (for an American style of mathematics), we have to deal with the following inconsistencies:

- ☐ Symbols from the Latin alphabet are typeset in math italic by default, and they can be switched to upright roman (or other alphabets).
- ☐ Symbols from the uppercase Greek alphabet are in roman by default, and they can be switched to math italic (or other alphabets).
- ☐ Symbols from the lowercase Greek alphabet are in math italic by default, but cannot switch families, because they are not available otherwise.

In addition, we have to deal with the following limitations of fonts:

- A bold math italic font (suitable for vectors) does exist in the standard set of Computer Modern fonts, but is not available as a math alphabet by default.
- A bold sans serif italic font (suitable for tensors) does not exist in the standard set of Computer Modern fonts, but could be substituted using non-standard fonts.
- An upright Greek font (suitable for elementary particles) is not available in the Computer Modern fonts and cannot easily be substituted, unless you switch to a different set of fonts.

To develop a setup suitable for typesetting physics according to the rules, the following steps must be taken:

- Load additional math families for bold math italic (suitable for vectors) and bold sans serif italic (suitable for tensors).
- Redefine the math codes of uppercase Greek letters, so that they are typeset in math italic instead of upright roman by default.
- Redefine the math codes of lowercase Greek letters, so that they can switch families where possible (in a controlled way).
- Define font switching macros for vectors, tensors, particles, and units, which prevent lowercase Greek letters from switching to math alphabets where they do not exist.
- Define macros for the markup of mathematical constants and operators to be set in upright roman where possible.
- Define macros for additional math operators and specific notations needed in physics.

Loading math families and math alphabets

In the early years of the project, when LaTeX 2.09 was still being used, defining additional math families or font switches was not really supported by the LaTeX format, so it was impractical to do for authors of macro packages. Instead, clumsy workarounds were being used to get a bold math italic font by standard means, which involved switching to `\boldmath` from inside an `\hbox`.

It was only after the introduction of NFSS2 in 1993 and LaTeX 2_ε in 1994 that it became feasible for authors of math support packages to redefine math fonts as needed. Given the LaTeX 2_ε interface, loading additional math families for vectors or tensors is relatively straight-forward:

```
\DeclareSymbolFont{vectors}{OML}{cmm}{b}{it}
\DeclareSymbolFont{tensors}{OT1}{cmss}{bx}{it}
```

Similarly, font switching commands for math alphabets can be defined as follows:

```
\DeclareSymbolFontAlphabet{\mathvec} {vectors}
\DeclareSymbolFontAlphabet{\mathtens}{tensors}
```

The only problem with this setup might be that the font shape `cmss/bx/it` does not exist in CM fonts and is silently substituted by `cmss/bx/n`, which results in using upright bold sans serif for tensors as often done by publishers. (This would work better with LM fonts nowadays, but we did not have them in the 1990s.) Once these fonts and alphabets are loaded, it becomes a matter of redefining the math codes to get the symbols to appear in the proper shape by default.

Redefining math codes

Given the default setup of (LA)TEX, the math codes for uppercase Greek letters are defined to be of type `\mathalpha` (which means: allowed to switch families), using the `operators` symbol font (upright roman) by default. To get them to use the `letters` symbol font (math italic) by default, we have to redefine the math codes as follows:

```
\DeclareMathSymbol{\Gamma}  {\mathalpha}{letters}{"00}
\DeclareMathSymbol{\Delta}  {\mathalpha}{letters}{"01}
...
\DeclareMathSymbol{\Omega}  {\mathalpha}{letters}{"0A}
```

In a more sophisticated setup, we could even make the default font configurable by a package option, using a macro to select the appropriate font. Nowadays, many support packages for math fonts tend to provide such switching mechanisms (and sometimes they also define extra macros such as `\upGamma` or `\varGamma`), but in the mid-1990s this was not the case.

For lowercase Greek letters, the situation is somewhat different. By default, the math codes are defined to be of type `\mathord` (which means: *not* allowed to switch families), using the `letters` symbol font (math italic) by default. To get them to switch to the `vectors` symbol font (bold math italic), we have to redefine the math codes using the type `\mathalpha`:

```
\DeclareMathSymbol{\alpha}  {\mathalpha}{letters}{"0B}
\DeclareMathSymbol{\beta}   {\mathalpha}{letters}{"0C}
...
\DeclareMathSymbol{\varphi} {\mathalpha}{letters}{"27}
```

Defining font switching macros

Unfortunately, allowing lowercase Greek letters to switch families not only allows them to switch to families that exist (such as bold math italic), but also allows them to switch to families which do not exist (such as upright roman), potentially causing unexpected results.

In order to prevent such effects, we found it necessary to add some intelligence to font switching macros to check whether the argument of the font switch is a lowercase Greek letter or something else which does not require special treatment. The solution we came up with consists of the following code:

```
\newif\if@lowgreek  \newbox\@lowgreekbox

\def\@lowgreektest#1{\setbox\@lowgreekbox=\hbox{$%
  \global\@lowgreekfalse
  \ifnum\alpha>#1\else\ifnum\varphi<#1\else
  \global\@lowgreektrue\fi\fi
$}}
\def\@lowgreekswitch#1#2#3{\@lowgreektest{#1}%
  \if@lowgreek\def\next{#3}\else\def\next{#2}\fi\next{#1}}
```

In essence, we use a numeric comparison of math codes in `\@lowgreektest` to check if the argument is lowercase Greek (between `\alpha` and `\varphi`) and we use the result of this check in `\@lowgreekswitch` to choose between two alternative font commands to use for lowercase Greek letters or everything else. Given these macros, we can define font switching macros as follows:

```
\DeclareRobustCommand{\particle}[1]{%
  \@lowgreekswitch{#1}{\mathrm}{\mathnormal}}
\DeclareRobustCommand{\tens}[1]{%
  \@lowgreekswitch{#1}{\mathtens}{\mathvec}}
```

The effect should be that \particle will switch to \mathnormal (math italic) for lowercase Greek and to \mathrm (upright roman) for everything else. Similarly, \tens will switch to \mathvec (bold math italic) for lowercase Greek and to \mathtens (bold sans serif italic or upright) for everything else.

For vectors, no special font switching macros are needed, if the default style of bold math italic is used. However, if a publisher style prefers to use \mathbf (upright bold) for vectors, a similar switch would also be needed:

```
% if publisher insists on \mathbf (upright bold) for vectors
\DeclareRobustCommand{\vec}[1]{%
  \@lowgreekswitch{#1}{\mathbf}{\mathvec}}
% otherwise, if using \mathvec (bold math italic) for vectors
\let\vec=\mathvec
```

As will be obvious from this discussion, setting up the appropriate font families to satisfy the requirements of physics could be relatively straight-forward if all the fonts were math fonts providing the full range of Greek and Latin alphabets and not just text fonts providing only a subset. However, given the traditional setup of (LA)TEX and its limitations, such kinds of workarounds seem to be unavoidable to provide for font substitutions when symbols are not available.

Defining logical markup for physics

As shown above, we have redefined \vec as a font switch, thereby overwriting the default definition as a math accent in (LA)TEX. Following the principle of logical markup, we have chosen to use \vec and a number of similar macros to mark up the meaning of entities in physical notation consistently, regardless of how they will be represented depending on package options of a macro package.

In summary, we have defined macros such as \vec (vectors), \tens (tensors), \text (textual indices), \units (physical units), \chem (chemical elements), or \particle (elementary particles), as summarized in the following table.

markup	purpose	font	scope
none (default)	physical quantities	\mathnormal	Latin and Greek
\units	physical units	\mathrm	Latin mostly
\text	textual material	\mathrm	Latin only
\chem	chemical elements	\mathrm	Latin only
\particle	elementary particles	see above	Latin and Greek
\vec	vector quantities	see above	Latin and Greek
\tens	tensor quantities	see above	Latin and Greek

In many cases, these macros are simply implemented as font switches if no special provisions for font substitutions are needed. In other cases, they could be either defined as font switches or as math accents with different representations depending on the preferred style of the author or publisher.

When using such markup, it is important to ensure a certain level of discipline and consistency when typing formulas. For example, there may be a difference between typing \dot\vec{x}} and \vec{\dot{x}}, especially if font substitutions involving \@lowgreektest and \@lowgreekswitch are used.

In general, macros which are known to be math accents (such as \dot or \hat) should be applied on the outer level to macros which could be implemented either as math accents or font switches (such as \vec or \tens). In addition, such macros should usually be applied to individual symbols only, so you should make sure to type \vec{E} \times \vec{B} instead of taking shortcuts such as \vec{E \times B}.

Defining markup for mathematical constants and operators

Besides the requirements for physical notation discussed so far, there are also requirements for mathematical constants (such as e, i, π) and for mathematical operators (such as d, ∂, δ, Δ), which are supposed to be typeset in upright roman, if the rules for typesetting physics are to be implemented to the full extent.

In practice, however, these rules are often neglected or only partially implemented for various reasons. One obvious problem is the lack of suitable fonts, making it difficult or impossible to typeset lowercase Greek letters (such as δ) and certain other symbols (such as ∂) in upright roman.

Another problem is the question of markup, which requires careful attention by authors or editors to make sure that individual letters (such as d, e, i) are typeset in the proper font depending on the context where they are used.

In particular, not every occurrence of these letters represents a mathematical constant or operator which should be typeset in upright roman. It always depends on the context: d could also be used as a distance, e could be used as the charge of an electron, and i could be used as a component index. In such cases, the letters would be considered physical quantities and typeset in math italic.

It is only when dx is used as a differential operator, e^x is used as an exponential term, and i is used as the imaginary unit, that these letters would be considered mathematical constants or operators and typeset in upright roman.

Concerning suitable markup, there does not seem to be any agreement between authors of macro packages how these entities should be represented. Obviously, it would be inconvenient for authors to type `\mathrm` all over the place and this would also violate the principle of logical markup. On the other hand, defining the shortest possible macros (such as `\d`, `\e`, `\i`) conflict with standard TEX macros for the under-dot and the dotless-i in text mode. (Alternatively, one might prefer to use `\dd`, `\ee`, `\ii` for these macros, which are still easy enough to type, but will not cause conflicts with existing macros.)

In our case, we have indeed chosen the shortest possible macros, but using a slightly more sophisticated setup to limit the redefinitions to math mode while retaining the original definitions in text mode (assuming that `\@@d` and `\@@i` are used to capture the original meanings):

```
\DeclareRobustCommand{\d}{%
  \relax\ifmmode\mathrm{d}\else\expandafter\@@d\fi}
\DeclareRobustCommand{\e}{%
  \relax\ifmmode\mathrm{e}\else\error\fi}
\DeclareRobustCommand{\i}{%
  \relax\ifmmode\mathrm{i}\else\expandafter\@@i\fi}
```

Yet another approach might be to use explicit markup only for some letters which will be used in different contexts (such as e, i), while using a special setup of math codes to achieve global changes for other letters which will always be used in the same context (such as d). For example, if the latter is only used as a mathematical operator, its math codes might just as well be redefined globally to set it in upright roman by default (using the `operators` font), thereby eliminating the need for authors to apply any special markup:

```
\DeclareMathSymbol{d}{\mathalpha}{operators}{`d}
```

Using this approach, authors could simply type `$dx$` without being aware of the fine details, while getting the proper rendering dx instead of dx automatically. Obviously, the same approach could also be used to set the preferred shape of other symbols which are almost always used as math operators (such as ∂, δ, Δ), provided that suitable fonts are available.

Summary and conclusions on setting up math fonts

In the previous sections, we have discussed a low-level approach to set up (LA)TEX for typesetting physics, based on the state of the art of the mid-1990s. At that time, there were relatively few choices of math fonts available besides Computer Modern, Concrete, or Euler; and math font support packages for LATEX usually provided little else but the essential setup. When we later switched to the commercial MathTime font set, relatively little changed in the basic setup, except that we created a custom virtual font to take advantage of the upright Greek font in the MathTime Plus edition.

In recent years, many more choices of math fonts have become available (such as txfonts, pxfonts, fourier, mathdesign), often providing useful additions such as upright or bold fonts; and several math font support packages for LATEX have started to provide various switching options (such as slantedgreek or uprightgreek) to configure the preferred style of rendering uppercase and lowercase Greek letters. Unfortunately, there is no universal interface for all such packages, and the details of what is available and/or configurable still vary quite a lot.

On another front of development, several packages have appeared which deal with the specifics of notations, including a recently released isomath package, which acts as a meta-package on top of other packages. Unfortunately, none of these packages covers all the details we encountered in our projects, and there is still no comprehensive package for a physics environment.

Looking towards the future, many recent developments of new TEX engines have concentrated on providing support for Unicode and OpenType font technology, including support for OpenType math fonts. Given these developments, the traditional concept of alphabetic symbols, which may switch between different font families, has to be reconsidered fundamentally.

In a traditional 8-bit TEX engine, multiple math fonts are loaded into different font families, each using the same slots for Latin or Greek alphabets to be rendered in different styles (such as roman, italic, bold, bold italic, script, etc.). In a Unicode setup, however, there will be only a single math font, using a different range of slots for each style of alphabets, so the font switching commands would actually have to switch between different slots, as shown in the following table:

font style	Latin alphabet slots	Greek alphabet slots
upright roman	U+0041 ... U+005A	U+0391 ... U+03A9
	U+0061 ... U+007A	U+03B1 ... U+03C9
math italic	U+1D400 ... U+1D433	U+1D6A8 ... U+1D6E1
upright bold	U+1D434 ... U+1D467	U+1D6E2 ... U+1D71B
bold math italic	U+1D468 ... U+1D49B	U+1D71C ... U+1D755
bold sans upright	U+1D5D4 ... U+1D607	U+1D756 ... U+1D79F
bold sans italic	U+1D63C ... U+1D66F	U+1D790 ... U+1D7C9

On the technical side, this method of switching alphabetic symbols between different Unicode slots will cause a lot of overhead to the implementation of font switching macros in support packages such as unicode-math, but fortunately these macros will have to be implemented only once and for all, since the same layout will be applicable to all forthcoming OpenType math fonts.

Regarding the range of available alphabets, Unicode math provides not only the above-mentioned alphabets, but also several more (including script, bold script, fraktur, bold fraktur, or blackboard bold). Moreover, the range of symbols includes not only the full uppercase and lowercase Latin and Greek alphabets, but also some letter-like operators such as ∇ and ∂. Given all these provisions, developing a full setup for typesetting physics should finally become much easier.

Improving the appearance of math formulas

Many users of TEX in academic research have spent little time on learning TEX, relying only on introductory (LA)TEX books or popular online guides such as lkurz or lshort. While these guides are useful to get started quickly, they usually do not spend much room on explaining the fine points of typesetting math. By contrast, *The TEXbook* devotes four chapters on the topic of math typesetting including a full chapter on the fine points of typing math formulas.

Before making any attempts to improve the appearance of math formulas, it is important to make sure that formulas are properly coded, and it may be worthwhile to refresh your reading of *The TEXbook* for this purpose.

Avoiding common mistakes

Some common mistakes by inexperienced users include forgetting to type the proper symbol (such as using <= or << instead of `\le` or `\ll` for $\leq$ or $\ll$) or failing to note the difference between similar symbols (such as using < and > instead of `\langle` and `\rangle` as delimiters for $\langle x \rangle$).

Another common mistake is forgetting to define suitable macros when special notations are needed. For example, in vector analysis, operators for the gradient, divergence, and rotation (curl) of a vector field may be defined as follows:

```
\def\grad{\mathop{\operator@font grad}\nolimits}
\def\div{\mathop{\operator@font div}\nolimits}
\def\rot{\mathop{\operator@font rot}\nolimits}
```

Simply typing `\mathrm{div}` or even `\mbox{div}` instead of using a `\mathop` may appear to give similar results, but will not produce the proper spacing.

Alignment of indices

By default, TEX uses a different shift amount for the placement of subscripts when a subscript appears by itself (such as in x_0) or when a subscript appears together with a superscript (such as in x'_0). While each case may be perfectly fine by itself, an inconsistency becomes apparent when both cases appear in the same formula, as in the example of a simple coordinate transform:

$$x(t) = x_0 + v_0 t \,. \qquad x'(t) = x'_0 + v'_0 t \,.$$

To avoid this kind of inconsistency there are two possible solutions (besides hacking the fontdimen parameters of math fonts, which may be a questionable solution). One solution consists of adding empty groups as phantom superscripts by typing `x_{0}^{}` to get x_0. In this case, all subscripts would be lowered a little bit more, as if a superscript was always present:

$$x(t) = x_0 + v_0 t \,, \qquad x'(t) = x'_0 + v'_0 t \,.$$

Another solution consists of inserting empty groups to avoid a build-up of superscripts and subscripts by typing `x'{}_{0}` to get x'_0. In this case, all subscripts would be lowered a little bit less, as if each appeared by itself without a superscript. Unfortunately, some backspacing will be needed to close the visual gaps, so you would have to type `x'{}_{\!\!\!0}` to get the following result:

$$x(t) = x_0 + v_0 t \,, \qquad x'(t) = x'_0 + v'_0 t \,.$$

In general, the first solution may be easier to use in displays, but the second one may be useful if you want to prevent an expression from becoming too big.

Sizes of delimiters

By default, TEX automatically determines the required size of big delimiters if `\left` and `\right` are used around a subformula. However, there are situations where `\left` and `\right` cannot be used, and there are also some situations where you may want to adjust the size of delimiters manually. In a typical setup, TEX provides the macros `\big`, `\Big`, `\bigg`, and `\Bigg`, which can be used to select specific sizes of delimiters, such as 12 pt, 18 pt, 24 pt, and 30 pt.

One example of using explicit font sizes is to avoid inconsistencies which may occur using the default settings. Consider the following equation:

$$\left(\frac{\cos\alpha}{\alpha}\right)^2+\left(\frac{\sin\beta}{\beta}\right)^2$$

As it turns out, the fraction involving $\cos\alpha$ is surrounded by 18 pt (Big) delimiters whereas the fraction involving $\sin\beta$ requires 24 pt (bigg) delimiters. The reason is simply that $\cos\alpha$ is smaller because it has no ascenders or descenders, whereas $\sin\beta$ is bigger because it has both. If you want to ensure a consistent size of delimiters, it may be preferable to use `\bigg` (24 pt) on both expressions:

$$\bigg(\frac{\cos\alpha}{\alpha}\bigg)^2+\bigg(\frac{\sin\beta}{\beta}\bigg)^2$$

Another example of using explicit sizes is to prevent delimiters from becoming too big. Consider the following equation:

$$\boldsymbol{R}=\left(\sum_{i=1}^{N} m_i\boldsymbol{r}_i\right)/M\,,\qquad M=\sum_{i=1}^{N} m_i\,.$$

Whenever you have delimiters around a summation the size calculation incorporates the upper and lower limits as well. If you just want to cover the summation sign without limits, it may be preferable to use `\Big` (18 pt) in this case:

$$\boldsymbol{R}=\Big(\sum_{i=1}^{N} m_i\boldsymbol{r}_i\Big)/M\,,\qquad M=\sum_{i=1}^{N} m_i\,.$$

Yet another example of using explicit sizes is to make delimiters bigger than they would normally appear. Consider the following equation:

$$\delta\int_{t_0}^{t_1} F\left(x(t),\dot{x}(t),t\right)=0\,.$$

Here, the outer level of delimiters is exactly the same size as the inner level, despite using `\left` and `\right` to get big delimiters. If you want the outer level to be more visible, it may be preferable to use `\big` (12 pt) in this case:

$$\delta\int_{t_0}^{t_1} F\Big(x(t),\dot{x}(t),t\Big)=0\,.$$

The same principle also applies if you have different kinds of nested delimiters and want the outer level to stand out, such as in this example:

$$\frac{\mathrm{d}\langle\widehat{\boldsymbol{p}}\rangle}{\mathrm{d}t}=\frac{1}{\mathrm{i}\hbar}\Big\langle[\widehat{\boldsymbol{p}},\widehat{H}]\Big\rangle\,.$$

Sizes of radicals

Unlike the size of delimiters which can be influenced manually, the size of radicals is always determined automatically based on the size of the subformula under the radical. In unfortunate circumstances, TEX may happen to choose a bigger size than you would like, such as in the following example:

$$\sqrt{p^2c^2+m_0^2c^4}\,, \qquad \frac{1}{\sqrt{p^2c^2+m_0^2c^4}}\,.$$

A tricky way to avoid this problem is to use staggered indices (as discussed earlier) to prevent subscripts from being lowered too much:

$$\sqrt{p^2c^2+{m_0}^2c^4}\,, \qquad \frac{1}{\sqrt{p^2c^2+{m_0}^2c^4}}\,.$$

Whether or not this problem occurs strongly depends on the choice of fonts and the relative size of indices used with these fonts.

When using a font family which provides optical design sizes, a typical setup will use font sizes of 10 pt / 7 pt / 5 pt for the text font and the first and second level indices. In this case, the 7 pt indices are usually small enough, so that a build-up of superscripts and subscripts does not exceed the body size.

When using a font without optical design sizes, the second level indices may become unreadable if a scaled-down version of the base font is used at 5 pt, so a different progression of sizes is used, such as 10 pt / 7.5 pt / 6 pt. In this case, the 7.5 pt indices may turn out a little too big, so that a build-up of superscripts and subscripts may require the next larger sizes of roots or delimiters.

Spacing and backspacing

By default, TEX does a good job of spacing in math mode based on the various classes of math symbols such as ordinary symbols, binary operators, relations, openings, closings, or punctuation. However, in some cases, it may be desirable or even required to insert space manually for better results.

The most common example of manual spacing occurs in integrals, where a thinspace (`\,`) is usually inserted before each differential term, such as in:

$$\int F(x,y,z)\,\mathrm{d}^3V\,, \qquad \int F(r,\vartheta,\varphi)\,r^2\mathrm{d}r\,\sin\vartheta\,\mathrm{d}\vartheta\,\mathrm{d}\varphi\,.$$

(There is no need for a thinspace after an index or exponent (such as in $r^2\mathrm{d}r$) where a visual gap may not be needed or before an operator (such as before $\sin\vartheta$) where space is inserted automatically. However, there is a need for a thinspace before $\mathrm{d}\vartheta$ and $\mathrm{d}\varphi$ and also before $r^2\mathrm{d}r$.)

In addition, it is also a good idea to insert manual spacing before punctuation in displayed equations (as already shown in many examples), and in places where you want to emphasize the logical structure, such as after a prefix term or before an exponential term, as shown in the following example:

$$\psi(x,t) = \int A(k)\,\mathrm{e}^{\mathrm{i}(kx-\omega(k)t)}\,\mathrm{d}k\,.$$

Besides such examples of manual spacing used to emphasize the logical structure, there are also situations where manual spacing is needed to avoid visual collisions or where

manual backspacing is needed to avoid visual gaps (such as in v^2/c^2). Some typical examples are explained in *The TEXbook*.

Another typical example of manual backspacing occurs when exponents are attached to big delimiters, as shown in the following example:

$$\mathrm{i}\hbar\frac{\partial\psi}{\partial t} = \frac{1}{2m}\left(\frac{\hbar}{\mathrm{i}}\boldsymbol{\nabla} - q\boldsymbol{A}\right)^2\psi\,.$$

Since TEX considers font metrics in terms of rectangular boxes and does not use an italic correction in this situation, the placement of superscripts is based only on the size of the box and does not take into account the rounded shape. To improve the appearance, it may be helpful to insert manual backspacing in the exponent by typing `\right)^{\!\!2}`, arriving at this result:

$$\mathrm{i}\hbar\frac{\partial\psi}{\partial t} = \frac{1}{2m}\left(\frac{\hbar}{\mathrm{i}}\boldsymbol{\nabla} - q\boldsymbol{A}\right)^{\!2}\psi\,.$$

Obviously, such corrections only apply to rounded or angular shapes, whereas square brackets do not need such corrections.

Finally, another interesting example of manual spacing or backspacing is the use of staggered indices in tensor analysis, such as in:

$$g^{\lambda\mu}g_{\mu\nu} = \delta^{\lambda}{}_{\nu}\,, \qquad g_{\lambda\mu}g^{\mu\nu} = \delta_{\lambda}{}^{\nu}\,.$$

Just inserting an empty group between superscripts and subscripts may create visual gaps, which may be compensated by a little bit of backspacing:

$$g^{\lambda\mu}g_{\mu\nu} = \delta^{\lambda}{}_{\nu}\,, \qquad g_{\lambda\mu}g^{\mu\nu} = \delta_{\lambda}{}^{\nu}\,.$$

Howe much backspacing is needed may depend on the individual letters involved, as the diagonal shape of λ and ν happens to coincide in this particular example.

Summary and conclusions on fine-tuning math formulas

In the previous sections, we have presented several examples of math formulas, illustrating various kinds of problems that may benefit from manual adjustments in order to improve the appearance of math formulas. Drawing from 2500 pages of material and seven years of experience, lots of examples and variety of notations have been seen, which are impossible to cover in this paper, but hopefully our examples may give helpful advice to authors involved in similar projects.

References

[1] Eckhard Rebhan: *Theoretische Physik I: Mechanik, Elektrodynamik, Spezielle und Allgemeine Relativititätstheorie, Kosmologie.*
Spektrum Akademischer Verlag, Heidelberg, 1999. xxvi + 1222 pp., ISBN 3-8274-0246-8

[2] Eckhard Rebhan: *Theoretische Physik II: Quantenmechanik, Quantenfeldtheorie, Elementarteilchen, Thermodynamik und Statistik.*
Spektrum Akademischer Verlag, Heidelberg, 2004. xxii + 1354 pp., ISBN 3-8274-0247-6

[3] Eckhard Rebhan: *Theoretische Physik: Mechanik.*
Spektrum Akademischer Verlag, Heidelberg, 2006. xiv + 390 pp., ISBN 3-8274-1716-3

[4] Eckhard Rebhan: *Theoretische Physik: Elektrodynamik.*
Spektrum Akademischer Verlag, Heidelberg, 2007. xii + 406 pp., ISBN 3-8274-1717-1

[5] Eckhard Rebhan: *Theoretische Physik: Quantenmechanik.*
Spektrum Akademischer Verlag, Heidelberg, 2008. xii + 536 pp., ISBN 3-8274-1718-3

[6] Eckhard Rebhan: *Theoretische Physik: Thermodynamik.*
Spektrum Akademischer Verlag, Heidelberg, 2009 (to appear). ISBN 3-8274-1719-X

[7] University of Chicago Press: *The Chicago Manual of Style.*
University of Chicago Press, 15th revised edition, 2003. xvii + 956 pp., ISBN 0-226-10403-6
http://www.chicagomanualofstyle.org/

[8] Ellen Swanson: *Mathematics into Type.*
American Mathematical Society, updated edition, 1999. x + 102 pp., ISBN 0-8218-1961-5

[9] International Union of Pure and Applied Physics, S.U.N. Commission: *Symbols, Units, and Nomenclature in Physics.* Document U.I.P. 20 (1978), also published in *Physica A*, 93:1–60, 1978.

[10] International Union of Pure and Applied Physics, SUNAMCO Commission: *Symbols, Units, Nomenclature and Fundamental Constants in Physics.* Document IUPAP 25 (1987), SUNAMCO 87-1, also published in *Physica A*, 146:1–67, 1987.

[11] International Union of Pure and Applied Chemistry, Physical Chemistry Division: *Quantities, Units, and Symbols in Physical Chemistry.*
Blackwell Science, 1st edition, 1988.

[12] International Union of Pure and Applied Chemistry, Physical Chemistry Division: *Quantities, Units, and Symbols in Physical Chemistry.*
Blackwell Science, 2nd edition, 1993.
http://old.iupac.org/publications/books/gbook/green_book_2ed.pdf

[13] International Union of Pure and Applied Chemistry, Physical Chemistry Division: *Quantities, Units, and Symbols in Physical Chemistry.*
Royal Society of Chemistry, 3rd edition, 2007.

[14] International Organization for Standardization: *Quantities and Units — Part 11: Mathematical signs and symbols for use in the physical sciences and technology.* Technical Report ISO 31-11:1992.
International Organization for Standardization, Geneva, 1992.

[15] International Organization for Standardization: *Quantities and Units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology.* Technical Report ISO 80000-2:2009.
International Organization for Standardization, Geneva, 2009.

[16] Claudio Beccari: *Typesetting mathematics for science and technology according to ISO 31-11.*
TUGboat, 18(1):39–48, March 1997.
http://www.tug.org/TUGboat/Articles/tb18-1/tb54becc.pdf

[17] Ulrik Vieth: *Requirements for Typesetting Physics.*
Math Font Group discussion document. Unpublished. October 1997.
http://www.tug.org/twg/mfg/papers/ulrik/physreq.pdf

Dr. Ulrik Vieth
Vaihinger Straße 69
70567 Stuttgart
Germany
ulrik dot vieth (at) arcor dot de