Experimental Unicode mathematical typesetting: The unicode-math package

Will Robertson
Philipp Stephani, Joseph Wright, Khaled Hosny
http://github.com/wspr/unicode-math

2017/11/18 v0.8i

Abstract

This document describes the unicode-math package, which is intended as an implementation of Unicode maths for LaTeX using the XeTeX and LuaTeX typesetting engines. With this package, changing maths fonts is as easy as changing text fonts — and there are more and more maths fonts appearing now. Maths input can also be simplified with Unicode since literal glyphs may be entered instead of control sequences in your document source.

The package provides support for both XeTeX and LuaTeX. The different engines provide differing levels of support for Unicode maths. Please let us know of any troubles.

Alongside this documentation file, you should be able to find a minimal example demonstrating the use of the package, 'unimath-example.ltx'. It also comes with a separate document, 'unimath-symbols.pdf', containing a complete listing of mathematical symbols defined by unicode-math, including comparisons between different fonts.

Finally, while the STIX fonts may be used with this package, accessing their alphabets in their 'private user area' is not yet supported. (Of these additional alphabets there is a separate caligraphic design distinct to the script design already included.) Better support for the STIX fonts is planned for an upcoming revision of the package after any problems have been ironed out with the initial version.

Contents

1 Introduction

This document describes the `unicode-math` package, which is an *experimental* implementation of a macro to Unicode glyph encoding for mathematical characters.

Users who desire to specify maths alphabets only (Greek and Latin letters, and Arabic numerals) may wish to use Andrew Moschou's `mathspec` package instead. (XeTeX-only at time of writing.)

2 Acknowledgements

Many thanks to: Microsoft for developing the mathematics extension to OpenType as part of Microsoft Office 2007; Jonathan Kew for implementing Unicode math support in XeTeX; Taco Hoekwater for implementing Unicode math support in LuaTeX; Barbara Beeton for her prodigious effort compiling the definitive list of Unicode math glyphs and their LaTeX names (inventing them where necessary), and also for her thoughtful replies to my sometimes incessant questions; Philipp Stephani for extending the package to support LuaTeX. Ross Moore and Chris Rowley have provided moral and technical support from the very early days with great insight into the issues we face trying to extend and use TeX in the future. Apostolos Syropoulos, Joel Salomon, Khaled Hosny, and Mariusz Wodzicki have been fantastic beta testers.

3 Getting started

Load `unicode-math` as a regular LaTeX package. It should be loaded after any other maths or font-related package in case it needs to overwrite their definitions. Here's an example using the filename syntax to load the TeX Gyre Pagella Math font: (this works for both XeLaTeX and LuaLaTeX)

```
\usepackage{amsmath} % if desired
\usepackage{unicode-math}
\setmathfont{texgyrepagella-math.otf}
```

Once the package is loaded, traditional TFM-based maths fonts are no longer supported; you can only switch to a different OpenType maths font using the `\setmathfont` command. If you do not load an OpenType maths font before `\begin{document}`, Latin Modern Math (see above) will be loaded automatically.

3.1 New commands

LaTeX, since the first version of LaTeX 2_ε, changed the math group selection from, say, `{\bf x}` to `\mathbf{x}`. It introduced commands such as `\mathbf`, `\mathit`, `\mathsf`, `\mathtt` and `\mathcal`, besides `\mathnormal`. This was not only done to maintain the analogy with `\textbf`, `\textit` and so on, but with the precise purpose of loading the needed math groups (or math families) on demand and not allocating them if not required by the document.

The introduction of unicode-math posed new problems. For instance, there is a big difference between say `fit` as an operator name in boldface type and the product of three boldface variables. With legacy TeX engines, `\mathbf{fit}` would use a ligature and the same would happen with the input `\mathbf{f}\mathbf{i}\mathbf{t}`. For the latter case, the user should probably use `\mathbf{f\/}`.

However, there is another important point from a *conceptual* point of view. A boldface variable name should be printed using the *math font*, whereas a boldface operator name should be printed using the *text font*. OpenType math fonts make this distinction feasible, because they contain several math alphabets. Of course a boldface text 'x' will not differ much (or at all) from a boldface math 'x', but this is not the point: they *should* be considered different, because the former is U+0078 in Unicode, the latter is U+1D431.

When discussing the matter, it soon appeared clear that *two* different commands are needed: one for using text boldface in math, one for using math boldface. Only the document's author can know whether one or the other is needed. The decision was to split off the two meanings by using either `\mathbf` (for the boldface text font in math) or `\symbf` (for the bold math font).

To be more detailed, most alphabet commands are provided with the `\math...` prefix synonyms, but there are five 'legacy' font alphabets that intentionally behave somewhat different. These are `\mathup`, `\mathit`, `\mathbf`, `\mathsf`, and `\mathtt`. (N.B.: `\mathrm` is defined as a synonym for `\mathup`, but the latter is prefered as it is a script-agnostic term.)

The `\symbf` command switches to single-letter mathematical symbols (generally within the same OpenType font). The `\mathbf` command switches to a text font that is set up to behave correctly in mathematics, and should be used for multi-letter identifiers. These could be denoted 'text math alphabets'; further details are discussed in section §4.4. Additional similar 'text math alphabet' commands can be defined using the `\setmathfontface` command discussed in section §4.4. To control the behaviour of the default text math alphabet commands to behave in a backwards-compatible mode, see the package options described in section §4.4.2.

Thus unicode-math provides a number of commands (such as `\symbfsf`) to select specific 'symbol alphabets' within the unicode maths font, with usage, e.g., `$\symbfsf{g}$` → **g**. The full listing is shown in Table 1. For backwards compatibility, many of these are also defined with 'familiar' synonyms such as `\mathbfsf`. However, where possible the 'sym' prefix commands should be preferred, as certain synonyms may become deprecated in time. The `\symliteral` command is described in section §5.1.

3.2 *Package options*

Package options may be set when the package as loaded or at any later stage with the `\unimathsetup` command. Therefore, the following two examples are equivalent:

```
\usepackage[math-style=TeX]{unicode-math}
% OR
```

Table 1: New unicode-math commands.

unicode-math command	Synonym
\symup	
\symit	
\symbf	
\symsf	
\symtt	
\symnormal	\mathnormal
\symliteral	
\symbfup	\mathbfup
\symbfit	\mathbfit
\symsfup	\mathsfup
\symsfit	\mathsfit
\symbfsfup	\mathbfsfup
\symbfsfit	\mathbfsfit
\symbfsf	\mathbfsf
\symbb	\mathbb
\symbbit	\mathbbit
\symscr	\mathscr
\symbfscr	\mathbfscr
\symcal	\mathcal
\symbfcal	\mathbfcal
\symfrak	\mathfrak
\symbffrak	\mathbffrak

Table 2: Package options.

Option	Description	See…
`math-style`	Style of letters	section §5.1
`bold-style`	Style of bold letters	section §5.2
`sans-style`	Style of sans serif letters	section §5.3
`nabla`	Style of the nabla symbol	section §5.5.1
`partial`	Style of the partial symbol	section §5.5.2
`colon`	Behaviour of `\colon`	section §5.5.5
`slash-delimiter`	Glyph to use for 'stretchy' slash	section §5.5.6

```
\usepackage{unicode-math}
\unimathsetup{math-style=TeX}
```

Note, however, that some package options affects how maths is initialised and changing an option such as `math-style` will not take effect until a new maths font is set up.

Package options may *also* be used when declaring new maths fonts, passed via options to the `\setmathfont` command. Therefore, the following two examples are equivalent:

```
\unimathsetup{math-style=TeX}
\setmathfont{Cambria Math}
% OR
\setmathfont{Cambria Math}[math-style=TeX]
```

A summary list of package options is shown in table 2. See following sections for more information.

4 *Unicode maths font setup*

In the ideal case, a single Unicode font will contain all maths glyphs we need. The file `unicode-math-table.tex` (based on Barbara Beeton's STIX table) provides the mapping between Unicode maths glyphs and macro names (all 3298 — or however many — of them!). A single command

`\setmathfont{`⟨*font name*⟩`}[`⟨*font features*⟩`]`

implements this for every every symbol and alphabetic variant. That means `x` to x, `\xi` to ξ, `\leq` to $\leq$, etc., `\symscr{H}` to $\mathscr{H}$ and so on, all for Unicode glyphs within a single font.

This package deals well with Unicode characters for maths input. This includes using literal Greek letters in formulae, resolving to upright or italic depending on preference.

Font features specific to `unicode-math` are shown in table 3. Package options (see table 2) may also be used. Other `fontspec` features are also valid.

Table 3: Maths font options.

Option	Description	See…
`range`	Style of letters	section §4.1
`script-font`	Font to use for sub- and super-scripts	section §4.2
`script-features`	Font features for sub- and super-scripts	section §4.2
`sscript-font`	Font to use for nested sub- and super-scripts	section §4.2
`sscript-features`	Font features for nested sub- and super-scripts	section §4.2

4.1 *Using multiple fonts*

There will probably be few cases where a single Unicode maths font suffices (simply due to glyph coverage). The STIX font comes to mind as a possible exception. It will therefore be necessary to delegate specific Unicode ranges of glyphs to separate fonts:

`\setmathfont{`⟨*font name*⟩`}[range=`⟨*unicode range*⟩`,`⟨*font features*⟩`]`

where ⟨*unicode range*⟩ is a comma-separated list of Unicode slot numbers and ranges such as `{"27D0-"27EB,"27FF,"295B-"297F}`. Note that TeX's syntax for accessing the slot number of a character, such as `` `\+ ``, will also work here.

You may also use the macro for accessing the glyph, such as `\int`, or whole collection of symbols with the same math type, such as `\mathopen`, or complete math styles such as `\symbb`. (Only numerical slots, however, can be used in ranged declarations.)

4.1.1 *Control over alphabet ranges*

As discussed earlier, Unicode mathematics consists of a number of 'alphabet styles' within a single font. In `unicode-math`, these ranges are indicated with the following (hopefully self-explanatory) labels:

```
up , it , tt , bfup , bfit , bb , bbit , scr , bfscr , cal , bfcal ,
frak , bffrak , sfup , sfit , bfsfup , bfsfit , bfsf
```

Fonts can be selected for specified ranges only using the following syntax, in which case all other maths font setup remains untouched:

- `[range=bb]` to use the font for 'bb' letters only.
- `[range=bfsfit/{greek,Greek}]` for Greek lowercase and uppercase only (also with `latin`, `Latin`, `num` as possible options for Latin lower-/upper-case and numbers, resp.).
- `[range=up->sfup]` to map to different output styles.

Note that 'meta-styles' such as '`bf`' and '`sf`' are not included here since they are context dependent. Use `[range=bfup]` and `[range=bfit]` to effect changes to the particular ranges selected by '`bf`' (and similarly for '`sf`').

If a particular math style is not defined in the font, we fall back onto the lower-base plane (i.e., 'upright') glyphs. Therefore, to use an ASCII-encoded fractur font, for example, write

```
\setmathfont{SomeFracturFont}[range=frak]
```

and because the math plane fractur glyphs will be missing, `unicode-math` will know to use the ASCII ones instead. If necessary this behaviour can be forced with `[range=frak->up]`, since the 'up' range corresponds to ASCII letters.

Users of the impressive Minion Math fonts (commercial) may use remapping to access the bold glyphs using:

```
\setmathfont{MinionMath-Regular.otf}
\setmathfont{MinionMath-Bold.otf}[range={bfup->up,bfit->it}]
```

To set up the complete range of optical sizes for these fonts, a font declaration such as the following may be used: (adjust may be desired according to the font size of the document)

```
\setmathfont{Minion Math}[
 SizeFeatures = {
  {Size =      -6.01,  Font = MinionMath-Tiny},
  {Size =  6.01-8.41,  Font = MinionMath-Capt},
  {Size =  8.41-13.01, Font = MinionMath-Regular},
  {Size = 13.01-19.91, Font = MinionMath-Subh},
  {Size = 19.91-,      Font = MinionMath-Disp}
 }]

\setmathfont{Minion Math}[range = {bfup->up,bfit->it},
 SizeFeatures = {
  {Size =      -6.01,  Font = MinionMath-BoldTiny},
  {Size =  6.01-8.41,  Font = MinionMath-BoldCapt},
  {Size =  8.41-13.01, Font = MinionMath-Bold},
  {Size = 13.01-19.91, Font = MinionMath-BoldSubh},
  {Size = 19.91-,      Font = MinionMath-BoldDisp}
 }]
```

v0.8: Note that in previous versions of `unicode-math`, these features were labelled `[range=\mathbb]` and so on. This old syntax is still supported for backwards compatibility, but is now discouraged.

4.2 *Script and scriptscript fonts/features*

Cambria Math uses OpenType font features to activate smaller optical sizes for scriptsize and scriptscriptsize symbols (the B and C, respectively, in A_{B_C}). Other typefaces (such as Minion Math) may use entirely separate font files.

The features `script-font` and `sscript-font` allow alternate fonts to be selected for the script and scriptscript sizes, and `script-features` and `sscript-features` to apply different OpenType features to them.

By default `script-features` is defined as `Style=MathScript` and `sscript-features` is `Style=MathScriptScript`. These correspond to the two levels of OpenType's `ssty` feature tag. If the `(s)script-features` options are specified manually, you must additionally specify the `Style` options as above.

4.3 Maths 'versions'

LaTeX uses a concept known as 'maths versions' to switch math fonts mid-document. This is useful because it is more efficient than loading a complete maths font from scratch every time—especially with thousands of glyphs in the case of Unicode maths! The canonical example for maths versions is to select a 'bold' maths font which might be suitable for section headings, say. (Not everyone agrees with this typesetting choice, though; be careful.)

To select a new maths font in a particular version, use the syntax

`\setmathfont{`⟨*font name*⟩`}[version=`⟨*version name*⟩`,`⟨*font features*⟩`]`

and to switch between maths versions mid-document use the standard LaTeX command `\mathversion{`⟨*version name*⟩`}`.

4.4 Legacy maths 'alphabet' commands

LaTeX traditionally uses `\DeclareMathAlphabet` and `\SetMathAlphabet` to define document commands such as `\mathit`, `\mathbf`, and so on. While these commands can still be used, unicode-math defines a wrapper command to assist with the creation of new such maths alphabet commands. This command is known as `\setmathface` in symmetry with fontspec's `\newfontface` command; it takes syntax:

`\setmathfontface`⟨*command*⟩`{`⟨*font name*⟩`}[`⟨*font features*⟩`]`

`\setmathfontface`⟨*command*⟩`{`⟨*font name*⟩`}[version=`⟨*version name*⟩`,`⟨*font features*⟩`]`

For example, if you want to define a new legacy maths alphabet font `\mathittt`:

```
\setmathfontface\mathittt{texgyrecursor-italic.otf}
...
$\mathittt{foo} = \mathittt{a} + \mathittt{b}$
```

4.4.1 Default 'text math' fonts

The five 'text math' fonts, discussed above, are: `\mathrm`, `\mathbf`, `\mathit`, `\mathsf`, and `\mathtt`. These commands are also defined with their original definition under synonyms `\mathtextrm`, `\mathtextbf`, and so on.

When selecting document fonts using fontspec commands such as `\setmainfont`, unicode-math inserts some additional code into fontspec that keeps the current default fonts 'in sync' with their corresponding `\mathrm` commands, etc.

For example, in standard LaTeX, `\mathsf` doesn't change even if the main document font is changed using `\renewcommand\sfdefault{...}`. With unicode-math loaded, after writing `\setsansfont{Helvetica}`, `\mathsf` will now be set in Helvetica.

Table 4: Maths text font configuration options. Note that `\mathup` and `\mathrm` are aliases of each other and cannot be configured separately.

Defaults (from 'text' font)	From 'maths symbols'
mathrm=text	mathrm=sym
mathup=text*	mathup=sym*
mathit=text	mathit=sym
mathsf=text	mathsf=sym
mathbf=text	mathbf=sym
mathtt=text	mathtt=sym

If the `\mathsf` font is set explicitly at any time in the preamble, this 'auto-following' does not occur. The legacy math font switches can be defined either with commands defined by fontspec (`\setmathrm`, `\setmathsf`, etc.) or using the more general `\setmathfontface\mathsf` interface defined by unicode-math.

4.4.2 *Replacing 'text math' fonts by symbols*

For certain types of documents that use legacy input syntax, it may be preferable to have `\mathbf` behave as if it were `\symbf` en masse (et cetera respectively). A series of package options (table 4) are provided to facilitate switching the definition of `\mathXYZ` for the five legacy text math font definitions.

For example, if in a particular document `\mathbf` is used only for choosing symbols of vectors and matrices, a dedicated symbol font (`\symbf`) will produce better spacing and will better match the main math font. In that case loading unicode-math with the `mathbf=sym` will achieve the desired result.

4.4.3 *Operator font*

LaTeX defines an internal command `\operator@font` for typesetting elements such as `\sin` and `\cos`. This font is selected from the legacy operators NFSS 'MathAlphabet', which is no longer relevant in the context of unicode-math. By default, the `\operator@font` command is defined to switch to the `\mathrm` font. You may now change these using the command:

```
\setoperatorfont\mathit
```

Or, to select a unicode-math range:

```
\setoperatorfont\symscr
```

For example, after the latter above, `$\sin x$` will produce '$\mathscr{sin}\, x$'.

5 *Maths input*

XeTeX's Unicode support allows maths input through two methods. Like classical TeX, macros such as `\alpha`, `\sum`, `\pm`, `\leq`, and so on, provide verbose access to the entire repertoire of characters defined by Unicode. The literal characters themselves may be used instead, for more readable input files.

5.1 *Math 'style'*

Classically, TeX uses italic lowercase Greek letters and *upright* uppercase Greek letters for variables in mathematics. This is contrary to the ISO standards of using italic forms for both upper- and lowercase. Furthermore, in various historical contexts, often associated with French typesetting, it was common to use upright uppercase *Latin* letters as well as upright upper- and lowercase Greek, but italic lowercase latin. Finally, it is not unknown to use upright letters for all characters, as seen in the Euler fonts.

The `unicode-math` package accommodates these possibilities with the option `math-style` that takes one of four (case sensitive) arguments: `TeX`, `ISO`, `french`, or `upright`.[1] The `math-style` options' effects are shown in brief in table 5.

The philosophy behind the interface to the mathematical symbols lies in LaTeX's attempt of separating content and formatting. Because input source text may come from a variety of places, the upright and 'mathematical' italic Latin and Greek alphabets are *unified* from the point of view of having a specified meaning in the source text. That is, to get a mathematical 'x', either the ASCII ('keyboard') letter `x` may be typed, or the actual Unicode character may be used. Similarly for Greek letters. The upright or italic forms are then chosen based on the `math-style` package option.

If glyphs are desired that do not map as per the package option (for example, an upright 'g' is desired but typing `$g$` yields 'g'), *markup* is required to specify this; to follow from the example: `\symup{g}`. Maths style commands such as `\symup` are detailed later.

'Literal' interface Some may not like this convention of normalising their input. For them, an upright `x` is an upright 'x' and that's that. (This will be the case when obtaining source text from copy/pasting PDF or Microsoft Word documents, for example.) For these users, the `literal` option to `math-style` will effect this behaviour. The `\symliteral{⟨syms⟩}` command can also be used, regardless of package setting, to force the style to match the literal input characters. This is a 'mirror' to `\symnormal{⟨syms⟩}` (also alias `\mathnormal`) which 'resets' the character mapping in its argument to that originally set up through package options.

5.2 *Bold style*

Similar as in the previous section, ISO standards differ somewhat to TeX's conventions (and classical typesetting) for 'boldness' in mathematics. In the past, it has been customary to use bold *upright* letters to denote things like vectors and matrices. For example, $\mathbf{M} = (M_x, M_y, M_z)$. Presumably, this was due to the relatively scarcity of bold italic fonts in the pre-digital typesetting era. It has been

[1] Interface inspired by Walter Schmidt's lucimatx package.

Table 5: Effects of the `math-style` package option.

	Example	
Package option	Latin	Greek
`math-style=ISO`	(a,z,B,X)	$(\alpha,\beta,\mathit{\Gamma},\mathit{\Xi})$
`math-style=TeX`	(a,z,B,X)	$(\alpha,\beta,\Gamma,\Xi)$
`math-style=french`	$(a,z,\mathrm{B},\mathrm{X})$	$(\upalpha,\upbeta,\Gamma,\Xi)$
`math-style=upright`	$(\mathrm{a},\mathrm{z},\mathrm{B},\mathrm{X})$	$(\upalpha,\upbeta,\Gamma,\Xi)$

Table 6: Effects of the `bold-style` package option.

	Example	
Package option	Latin	Greek
`bold-style=ISO`	$(\boldsymbol{a},\boldsymbol{z},\boldsymbol{B},\boldsymbol{X})$	$(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\mathit{\Gamma}},\boldsymbol{\mathit{\Xi}})$
`bold-style=TeX`	$(\mathbf{a},\mathbf{z},\mathbf{B},\mathbf{X})$	$(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{\Gamma},\mathbf{\Xi})$
`bold-style=upright`	$(\mathbf{a},\mathbf{z},\mathbf{B},\mathbf{X})$	$(\boldsymbol{\upalpha},\boldsymbol{\upbeta},\mathbf{\Gamma},\mathbf{\Xi})$

suggested by some that *italic* bold symbols should be used nowadays instead, but this practise is certainly not widespread.

Bold Greek letters have simply been bold variant glyphs of their regular weight, as in $\boldsymbol{\xi} = (\xi_r, \xi_\phi, \xi_\theta)$. Confusingly, the syntax in LaTeX traditionally has been different for obtaining 'normal' bold symbols in Latin and Greek: `\mathbf` in the former ('**M**'), and `\bm` (or `\boldsymbol`, deprecated) in the latter ('$\boldsymbol{\xi}$').

In `unicode-math`, the `\symbf` command works directly with both Greek and Latin maths characters and depending on package option either switches to upright for Latin letters (`bold-style=TeX`) as well or keeps them italic (`bold-style=ISO`). To match the package options for non-bold characters, with option `bold-style=upright` all bold characters are upright, and `bold-style=literal` does not change the upright/italic shape of the letter. The `bold-style` options' effects are shown in brief in table 6.

Upright and italic bold mathematical letters input as direct Unicode characters are normalised with the same rules. For example, with `bold-style=TeX`, a literal bold italic latin character will be typeset upright.

Note that `bold-style` is independent of `math-style`, although if the former is not specified then matching defaults are chosen based on the latter.

5.3 *Sans serif style*

Unicode contains upright and italic, medium and bold mathematical style characters. These may be explicitly selected with the `\mathsfup`, `\mathsfit`, `\mathbfsfup`, and `\mathbfsfit` commands discussed in section §5.4.

How should the generic `\mathsf` behave? Unlike bold, sans serif is used much more sparingly in mathematics. I've seen recommendations to typeset tensors in sans serif italic or sans serif italic bold (e.g., examples in the `isomath` and `mattens` packages). But LaTeX's `\mathsf` is *upright* sans serif.

Therefore I reluctantly add the package options `[sans-style=upright]` and `[sans-style=italic]` to control the behaviour of `\mathsf`. The `upright` style sets up the command to use upright sans serif, including Greek; the `italic` style switches to using italic in both Latin and Greek. In other words, this option simply changes the meaning of `\mathsf` to either `\mathsfup` or `\mathsfit`, respectively. Please let me know if more granular control is necessary here.

There is also a `[sans-style=literal]` setting, set automatically with `[math-style=literal]`, which retains the uprightness of the input characters used when selecting the sans serif output.

5.3.1 *What about bold sans serif?*

While you might want your bold upright and your sans serif italic, I don't believe you'd also want your bold sans serif upright (or all vice versa, if that's even conceivable). Therefore, bold sans serif follows from the setting for sans serif; it is completely independent of the setting for bold.

In other words, `\mathbfsf` is either `\mathbfsfup` or `\mathbfsfit` based on `[sans-style=upright]` or `[sans-style=italic]`, respectively. And `[sans-style = literal]` causes `\mathbfsf` to retain the same italic or upright shape as the input, and turns it bold sans serif.

N.B.: there is no medium-weight sans serif Greek range in Unicode. Therefore, `\symsf{\alpha}` does not make sense (it produces '𝛼'), while `\symbfsf{\alpha}` gives '𝝰' or '𝞪' according to the `sans-style`.

5.4 *All (the rest) of the mathematical styles*

Unicode contains separate codepoints for most if not all variations of style shape one may wish to use in mathematical notation. The complete list is shown in table 7. Some of these have been covered in the previous sections.

The math font switching commands do not nest; therefore if you want sans serif bold, you must write `\symbfsf{...}` rather than `\symbf{\symsf{...}}`. This may change in the future.

5.4.1 *Double-struck*

The double-struck style (also known as 'blackboard bold') consists of upright Latin letters {𝕒–𝕫,𝔸ℤ}, numerals 𝟘–𝟡, summation symbol ⅀, and four Greek letters only: {ℽℼℾℿ}.

While `\symbb{\sum}` does produce a double-struck summation symbol, its limits aren't properly aligned. Therefore, either the literal character or the control sequence `\Bbbsum` are recommended instead.

There are also five Latin *italic* double-struck letters: ⅅⅆⅇⅈⅉ. These can be accessed (if not with their literal characters or control sequences) with the `\mathbbit` style switch, but note that only those five letters will give the expected output.

Table 7: Mathematical styles defined in Unicode. Black dots indicate an style exists in the font specified; blue dots indicate shapes that should always be taken from the upright font even in the italic style. See main text for description of `\mathbbit`.

Font				Alphabet		
Style	Shape	Series	Switch	Latin	Greek	Numerals
Serif	Upright	Normal	`\mathup`	•	•	•
		Bold	`\mathbfup`	•	•	•
	Italic	Normal	`\mathit`	•	•	• (blue)
		Bold	`\mathbfit`	•	•	• (blue)
Sans serif	Upright	Normal	`\mathsfup`	•		•
	Italic	Normal	`\mathsfit`	•		• (blue)
	Upright	Bold	`\mathbfsfup`	•	•	•
	Italic	Bold	`\mathbfsfit`	•	•	• (blue)
Typewriter	Upright	Normal	`\mathtt`	•		•
Double-struck	Upright	Normal	`\mathbb`	•		•
	Italic	Normal	`\mathbbit`	•		
Script	Upright	Normal	`\mathscr`	•		
		Bold	`\mathbfscr`	•		
Fraktur	Upright	Normal	`\mathfrak`	•		
		Bold	`\mathbffrac`	•		

5.4.2 *Caligraphic vs. Script variants*

The Unicode maths encoding contains a style for 'Script' letters, and while by default `\mathcal` and `\mathscr` are synonyms, there are some situations when a separate 'Caligraphic' style is needed as well.

If a font contains alternate glyphs for a separat caligraphic style, they can be selected explicitly as shown below. This feature is currently only supported by the XITS Math font, where the caligraphic letters are accessed with the same glyph slots as the script letters but with the first stylistic set feature (`ss01`) applied.

```
\setmathfont{xits-math.otf}[range={cal,bfcal},StylisticSet=1]
```

An example is shown below.

The Script style (`\mathscr`) in XITS Math is: $\mathscr{ABCXYZ}$

The Caligraphic style (`\mathcal`) in XITS Math is: $\mathcal{ABCXYZ}$

5.5 *Miscellanea*

5.5.1 *Nabla*

The symbol ∇ comes in the six forms shown in table 8. We want an individual option to specify whether we want upright or italic nabla by default (when either upright or italic nabla is used in the source). TeX classically uses an upright nabla,

Table 8: The various forms of nabla.

Description		Glyph
Upright	Serif	∇
	Bold serif	$\mathbf{\nabla}$
	Bold sans	$\boldsymbol{\nabla}$
Italic	Serif	$\mathit{\nabla}$
	Bold serif	$\boldsymbol{\mathit{\nabla}}$
	Bold sans	$\boldsymbol{\mathit{\nabla}}$

Table 9: The partial differential.

Description		Glyph
Regular	Upright	∂
	Italic	∂
Bold	Upright	$\mathbf{\partial}$
	Italic	$\boldsymbol{\partial}$
Sans bold	Upright	$\mathbf{\partial}$
	Italic	$\boldsymbol{\partial}$

and ISO standards agree with this convention. The package options `nabla=upright` and `nabla=italic` switch between the two choices, and `nabla=literal` respects the shape of the input character. This is then inherited through `\symbf`; `\symit` and `\symup` can be used to force one way or the other.

`nabla=italic` is the default. `nabla=literal` is activated automatically after `math-style=literal`.

5.5.2 *Partial*

The same applies to the symbols U+2202 partial differential and U+1D715 math italic partial differential.

At time of writing, both the Cambria Math and STIX fonts display these two glyphs in the same italic style, but this is hopefully a bug that will be corrected in the future — the 'plain' partial differential should really have an upright shape.

Use the `partial=upright` or `partial=italic` package options to specify which one you would like, or `partial=literal` to have the same character used in the output as was used for the input. The default is (always, unless someone requests and argues otherwise) `partial=italic`.[2] `partial=literal` is activated following `math-style=literal`.

See table 9 for the variations on the partial differential symbol.

5.5.3 *Primes*

Primes (x') may be input in several ways. You may use any combination the ASCII straight quote (') or the Unicode prime U+2032 (′); when multiple primes occur next to each other, they chain together to form double, triple, or quadruple primes if the font contains pre-drawn glyphs. The individual prime glyphs are accessed, as usual, with the `\prime` command, and the double-, triple-, and quadruple-prime glyphs are available with `\dprime`, `\trprime`, and `\qprime`, respectively.

If the font does not contain the pre-drawn glyphs or more than four primes are used, the single prime glyph is used multiple times with a negative kern to get

[2] A good argument would revolve around some international standards body recommending upright over italic. I just don't have the time right now to look it up.

A $^{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ +\ -\ =\ (\ )\ i\ n\ n\ h\ j\ r\ w\ y}$ Z

Figure 1: The Unicode superscripts supported as input characters. These are the literal glyphs from Charis SIL, not the output seen when used for maths input. The 'A' and 'Z' are to provide context for the size and location of the superscript glyphs.

A $_{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ +\ -\ =\ (\ )\ a\ e\ i\ o\ r\ u\ v\ x\ \beta\ \gamma\ \rho\ \varphi\ \chi}$ Z

Figure 2: The Unicode subscripts supported as input characters. See note from figure 1.

the spacing right. There is no user interface to adjust this negative kern yet (because I haven't decided what it should look like); if you need to, write something like this:

```
\ExplSyntaxOn
\muskip_gset:Nn \g_@@_primekern_muskip { -\thinmuskip/2 }
\ExplySyntaxOff
```

Backwards or reverse primes behave in exactly the same way; use the ASCII back tick (`) or the Unicode reverse prime `U+2035` (‵). The command to access the backprime is `\backprime`, and multiple backwards primes can accessed with `\backdprime`, `\backtrprime`, and `\backqprime`.

In all cases above, no error checking is performed if you attempt to access a multi-prime glyph in a font that doesn't contain one. For this reason, it may be safer to write `x''''` instead of `x\qprime` in general.

If you ever need to enter the straight quote `'` or the backtick `` ` `` in maths mode, these glyphs can be accessed with `\mathstraightquote` and `\mathbacktick`.

5.5.4 *Unicode subscripts and superscripts*

You may, if you wish, use Unicode subscripts and superscripts in your source document. For basic expressions, the use of these characters can make the input more readable. Adjacent sub- or super-scripts will be concatenated into a single expression.

The range of subscripts and superscripts supported by this package are shown in figures 1 and 2. Please request more if you think it is appropriate.

5.5.5 *Colon*

The colon is one of the few confusing characters of Unicode maths. In TeX, `:` is defined as a colon with relation spacing: '$a : b$'. While `\colon` is defined as a colon with punctuation spacing: '$a\colon b$'.

In Unicode, `U+003A` colon is defined as a punctuation symbol, while `U+2236` ratio is the colon-like symbol used in mathematics to denote ratios and other things.

Table 10: Slashes and backslashes.

Slot	Name	Glyph	Command
U+002F	SOLIDUS	/	\slash
U+2044	FRACTION SLASH	⁄	\fracslash
U+2215	DIVISION SLASH	∕	\divslash
U+29F8	BIG SOLIDUS	⧸	\xsol
U+005C	REVERSE SOLIDUS	\	\backslash
U+2216	SET MINUS	∖	\smallsetminus
U+29F5	REVERSE SOLIDUS OPERATOR	⧵	\setminus
U+29F9	BIG REVERSE SOLIDUS	⧹	\xbsol

This breaks the usual straightforward mapping from control sequence to Unicode input character to (the same) Unicode glyph.

To preserve input compatibility, we remap the ASCII input character ':' to U+2236. Typing a literal U+2236 char will result in the same output. If amsmath is loaded, then the definition of \colon is inherited from there (it looks like a punctuation colon with additional space around it). Otherwise, \colon is made to output a colon with \mathpunct spacing.

The package option colon=literal forces ASCII input ':' to be printed as \mathcolon instead.

5.5.6 *Slashes and backslashes*

There are several slash-like symbols defined in Unicode. The complete list is shown in table 10.

In regular LaTeX we can write \left\slash…\right\backslash and so on and obtain extensible delimiter-like symbols. Not all of the Unicode slashes are suitable for this (and do not have the font support to do it).

Slash Of U+2044 fraction slash, TR25 says that it is:

> …used to build up simple fractions in running text…however parsers of mathematical texts should be prepared to handle fraction slash when it is received from other sources.

U+2215 division slash should be used when division is represented without a built-up fraction; $\pi \approx 22/7$, for example.

U+29F8 big solidus is a 'big operator' (like $\sum$).

Backslash The U+005C reverse solidus character \backslash is used for denoting double cosets: $A \backslash B$. (So I'm led to believe.) It may be used as a 'stretchy' delimiter if supported by the font.

MathML uses U+2216 set minus like this: $A \smallsetminus B$.[3] The LaTeX command name \smallsetminus is used for backwards compatibility.

Presumably, U+29F5 reverse solidus operator is intended to be used in a similar way, but it could also (perhaps?) be used to represent 'inverse division': $\pi \approx 7 \setminus 22$.[4] The LaTeX name for this character is \setminus.

Finally, U+29F9 big reverse solidus is a 'big operator' (like $\sum$).

How to use all of these things Unfortunately, font support for the above characters/glyphs is rather inconsistent. In Cambria Math, the only slash that grows (say when writing

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bigg/ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad)$$

is the FRACTION SLASH, which we just established above is sort of only supposed to be used in text.

Of the above characters, the following are allowed to be used after \left, \middle, and \right:

- \fracslash;
- \slash; and,
- \backslash (the only reverse slash).

However, we assume that there is only *one* stretchy slash in the font; this is assumed by default to be U+002F solidus. Writing \left/ or \left\slash or \left\fracslash will all result in the same stretchy delimiter being used.

The delimiter used can be changed with the slash-delimiter package option. Allowed values are ascii, frac, and div, corresponding to the respective Unicode slots.

For example: as mentioned above, Cambria Math's stretchy slash is U+2044 fraction slash. When using Cambria Math, then unicode-math should be loaded with the slash-delimiter=frac option. (This should be a font option rather than a package option, but it will change soon.)

5.5.7 *Growing and non-growing accents*

There are a few accents for which TeX has both non-growing and growing versions. Among these are \hat and \tilde; the corresponding growing versions are called \widehat and \widetilde, respectively.

Older versions of XeTeX and LuaTeX did not support this distinction, however, and *all* accents there were growing automatically. (I.e., \hat and \widehat are equivalent.) As of LuaTeX v0.65 and XeTeX v0.9998, these wide/non-wide commands will again behave in their expected manner.

[3] §4.4.5.11 http://www.w3.org/TR/MathML3/

[4] This is valid syntax in the Octave and Matlab programming languages, in which it means matrix inverse pre-multiplication. I.e., $A \setminus B \equiv A^{-1}B$.

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	\cdotp	·			
U+22C5	\cdot	⋅			
U+2219	\vysmblkcircle	∙	∘	\vysmwhtcircle	U+2218
U+2022	\smblkcircle	•	◦	\smwhtcircle	U+25E6
U+2981	\mdsmblkcircle	⦁	⚬	\mdsmwhtcircle	U+26AC
U+26AB	\mdblkcircle	⚫	⚪	\mdwhtcircle	U+26AA
U+25CF	\mdlgblkcircle	●	○	\mdlgwhtcircle	U+25CB
U+2B24	\lgblkcircle	⬤	◯	\lgwhtcircle	U+25EF

Table 11: Filled and hollow Unicode circles.

5.5.8 *Pre-drawn fraction characters*

Pre-drawn fractions U+00BC–U+00BE, U+2150–U+215E are not suitable for use in mathematics output. However, they can be useful as input characters to abbreviate common fractions.

¼ ½ ¾ ↉ ⅐ ⅑ ⅒ ⅓ ⅔ ⅕ ⅖ ⅗ ⅘ ⅙ ⅚ ⅛ ⅜ ⅝ ⅞

For example, instead of writing '\tfrac12 x', you may consider it more readable to have '½x' in the source instead.

If the \tfrac command exists (i.e., if amsmath is loaded or you have specially defined \tfrac for this purpose), it will be used to typeset the fractions. If not, regular \frac will be used. The command to use (\tfrac or \frac) can be forced either way with the package option active-frac=small or active-frac=normalsize, respectively.

5.5.9 *Circles*

Unicode defines a large number of different types of circles for a variety of mathematical purposes. There are thirteen alone just considering the all white and all black ones, shown in table 11.

LaTeX defines considerably fewer: \circ and \bigcirc for white; \bullet for black. This package maps those commands to \vysmwhtcircle, \mdlgwhtcircle, and \smblkcircle, respectively.

5.5.10 *Triangles*

While there aren't as many different sizes of triangle as there are circle, there's some important distinctions to make between a few similar characters. See table 12 for the full summary.

These triangles all have different intended meanings. Note for backwards compatibility with TeX, U+25B3 has *two* different mappings in unicode-math. \bigtriangleup is intended as a binary operator whereas \triangle is intended to be used as a letter-like symbol.

Slot	Command	Glyph	Class
U+25B5	\vartriangle	▵	binary
U+25B3	\bigtriangleup	△	binary
U+25B3	\triangle	△	ordinary
U+2206	\increment	∆	ordinary
U+0394	\mathup\Delta	Δ	ordinary

Table 12: Different upwards pointing triangles.

But you're better off if you're using the latter form to indicate an increment to use the glyph intended for this purpose, U+2206: Δx.

Finally, given that △ and ∆ are provided for you already, it is better off to only use upright Greek Delta Δ if you're actually using it as a symbolic entity such as a variable on its own.

6 *Advanced*

6.1 *Warning messages*

This package can produce a number of informational messages to try and inform the user when something might be going wrong due to package conflicts or something else. As an experimental feature, these can be turn off on an individual basis with the package option `warnings-off` which takes a comma-separated list of warnings to suppress. A warning will give you its name when printed on the console output; e.g.,

```
* unicode-math warning: "mathtools-colon"
*
* ... <warning message> ...
```

This warning could be suppressed by loading the package as follows:

```
\usepackage[warnings-off={mathtools-colon}]{unicode-math}
```

6.2 *Programmer's interface*

(Tentative and under construction.) If you are writing some code that needs to know the current maths style (`\mathbf`, `\mathit`, etc.), you can query the variable `\l_@@_mathstyle_tl`. It will contain the maths style without the leading 'math' string; for example, `\symbf { \show \l_@@_mathstyle_tl }` will produce 'bf'.

A STIX table data extraction

The source for the TeX names for the very large number of mathematical glyphs are provided via Barbara Beeton's table file for the STIX project (`ams.org/STIX`). A version is located at `http://www.ams.org/STIX/bnb/stix-tbl.asc` but check `http://www.ams.org/STIX/` for more up-to-date info.

This table is converted into a form suitable for reading by TeX. A single file is produced containing all (more than 3298) symbols. Future optimisations might include generating various (possibly overlapping) subsets so not all definitions must be read just to redefine a small range of symbols. Performance for now seems to be acceptable without such measures.

This file is currently developed outside this DTX file. It will be incorporated when the final version is ready. (I know this is not how things are supposed to work!)

B Documenting maths support in the NFSS

In the following, ⟨*NFSS decl.*⟩ stands for something like `{T1}{lmr}{m}{n}`.

Maths symbol fonts Fonts for symbols: $\propto$, $\leq$, $\rightarrow$

`\DeclareSymbolFont{`⟨*name*⟩`}`⟨*NFSS decl.*⟩
Declares a named maths font such as `operators` from which symbols are defined with `\DeclareMathSymbol`.

Maths alphabet fonts Fonts for $ABC - xyz$, $\mathfrak{ABC} - \mathcal{XYZ}$, etc.

`\DeclareMathAlphabet{`⟨*cmd*⟩`}`⟨*NFSS decl.*⟩

For commands such as `\mathbf`, accessed through maths mode that are unaffected by the current text font, and which are used for alphabetic symbols in the ASCII range.

`\DeclareSymbolFontAlphabet{`⟨*cmd*⟩`}{`⟨*name*⟩`}`

Alternative (and optimisation) for `\DeclareMathAlphabet` if a single font is being used for both alphabetic characters (as above) and symbols.

Maths 'versions' Different maths weights can be defined with the following, switched in text with the `\mathversion{`⟨*maths version*⟩`}` command.

`\SetSymbolFont{`⟨*name*⟩`}{`⟨*maths version*⟩`}`⟨*NFSS decl.*⟩
`\SetMathAlphabet{`⟨*cmd*⟩`}{`⟨*maths version*⟩`}`⟨*NFSS decl.*⟩

Maths symbols Symbol definitions in maths for both characters (=) and macros (`\eqdef`): `\DeclareMathSymbol{`⟨*symbol*⟩`}{`⟨*type*⟩`}{`⟨*named font*⟩`}{`⟨*slot*⟩`}` This is the macro that actually defines which font each symbol comes from and how they behave.

Delimiters and radicals use wrappers around TeX's `\delimiter`/`\radical` primitives, which are re-designed in XeTeX. The syntax used in LaTeX's NFSS is therefore not so relevant here.

Delimiters A special class of maths symbol which enlarge themselves in certain contexts.

`\DeclareMathDelimiter{`⟨*symbol*⟩`}{`⟨*type*⟩`}{`⟨*sym. font*⟩`}{`⟨*slot*⟩`}{`⟨*sym. font*⟩`}{`⟨*slot*⟩`}`

Radicals Similar to delimiters (`\DeclareMathRadical` takes the same syntax) but behave 'weirdly'.

In those cases, glyph slots in *two* symbol fonts are required; one for the small ('regular') case, the other for situations when the glyph is larger. This is not the case in XeTeX.

Accents are not included yet.

Summary For symbols, something like:

```
\def\DeclareMathSymbol#1#2#3#4{
  \global\mathchardef#1"\mathchar@type#2
    \expandafter\hexnumber@\csname sym#2\endcsname
    {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}
```

For characters, something like:

```
\def\DeclareMathSymbol#1#2#3#4{
  \global\mathcode`#1"\mathchar@type#2
    \expandafter\hexnumber@\csname sym#2\endcsname
    {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}}
```

C Legacy TeX font dimensions

Text fonts	
ϕ_1	slant per pt
ϕ_2	interword space
ϕ_3	interword stretch
ϕ_4	interword shrink
ϕ_5	x-height
ϕ_6	quad width
ϕ_7	extra space
ϕ_8	cap height (XeTeX only)

Maths font, `\fam2`	
σ_5	x height
σ_6	quad
σ_8	num1
σ_9	num2
σ_{10}	num3
σ_{11}	denom1
σ_{12}	denom2
σ_{13}	sup1
σ_{14}	sup2
σ_{15}	sup3
σ_{16}	sub1
σ_{17}	sub2
σ_{18}	sup drop
σ_{19}	sub drop
σ_{20}	delim1
σ_{21}	delim2
σ_{22}	axis height

Maths font, `\fam3`	
ξ_8	default rule thickness
ξ_9	big op spacing1
ξ_{10}	big op spacing2
ξ_{11}	big op spacing3
ξ_{12}	big op spacing4
ξ_{13}	big op spacing5

D XeTeX math font dimensions

These are the extended `\fontdimens` available for suitable fonts in XeTeX. Note that LuaTeX takes an alternative route, and this package will eventually provide a wrapper interface to the two (I hope).

`\fontdimen`	Dimension name	Description
10	ScriptPercentScaleDown	Percentage of scaling down for script level 1. Suggested value: 80%.
11	ScriptScriptPercentScale-Down	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.
12	DelimitedSubFormulaMin-Height	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height × 1.5.
13	DisplayOperatorMinHeight	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.

\fontdimen	Dimension name	Description
14	MathLeading	White space to be left between math formulas to ensure proper line spacing. For example, for applications that treat line gap as a part of line ascender, formulas with ink going above (os2.sTypoAscender + os2.sTypoLineGap – MathLeading) or with ink going below os2.sTypoDescender will result in increasing line height.
15	AxisHeight	Axis height of the font.
16	AccentBaseHeight	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font (os2.sxHeight) plus any possible overshots.
17	FlattenedAccentBase-Height	Maximum (ink) height of accent base that does not require flattening the accents. Suggested: cap height of the font (os2.sCapHeight).
18	SubscriptShiftDown	The standard shift down applied to subscript elements. Positive for moving in the downward direction. Suggested: os2.ySubscriptYOffset.
19	SubscriptTopMax	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: /5 x-height.
20	SubscriptBaselineDropMin	Minimum allowed drop of the baseline of subscripts relative to the (ink) bottom of the base. Checked for bases that are treated as a box or extended shape. Positive for subscript baseline dropped below the base bottom.
21	SuperscriptShiftUp	Standard shift up applied to superscript elements. Suggested: os2.ySuperscriptYOffset.
22	SuperscriptShiftUpCramped	Standard shift of superscripts relative to the base, in cramped style.
23	SuperscriptBottomMin	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: ¼ x-height.

\fontdimen	Dimension name	Description
24	SuperscriptBaselineDrop-Max	Maximum allowed drop of the baseline of superscripts relative to the (ink) top of the base. Checked for bases that are treated as a box or extended shape. Positive for superscript baseline below the base top.
25	SubSuperscriptGapMin	Minimum gap between the superscript and subscript ink. Suggested: 4×default rule thickness.
26	SuperscriptBottomMax-WithSubscript	The maximum level to which the (ink) bottom of superscript can be pushed to increase the gap between superscript and subscript, before subscript starts being moved down. Suggested: /5 x-height.
27	SpaceAfterScript	Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font.
28	UpperLimitGapMin	Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator.
29	UpperLimitBaselineRiseMin	Minimum distance between baseline of upper limit and (ink) top of the base operator.
30	LowerLimitGapMin	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	LowerLimitBaselineDrop-Min	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	StackTopShiftUp	Standard shift up applied to the top element of a stack.
33	StackTopDisplayStyleShift-Up	Standard shift up applied to the top element of a stack in display style.
34	StackBottomShiftDown	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
35	StackBottomDisplayStyle-ShiftDown	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	StackGapMin	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element. Suggested: 3×default rule thickness.

\fontdimen	Dimension name	Description
37	StackDisplayStyleGapMin	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: 7×default rule thickness.
38	StretchStackTopShiftUp	Standard shift up applied to the top element of the stretch stack.
39	StretchStackBottomShift-Down	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.
40	StretchStackGapAboveMin	Minimum gap between the ink of the stretched element, and the (ink) bottom of the element above. Suggested: UpperLimitGapMin
41	StretchStackGapBelowMin	Minimum gap between the ink of the stretched element, and the (ink) top of the element below. Suggested: LowerLimitGapMin.
42	FractionNumeratorShiftUp	Standard shift up applied to the numerator.
43	FractionNumerator-DisplayStyleShiftUp	Standard shift up applied to the numerator in display style. Suggested: StackTopDisplayStyleShiftUp.
44	FractionDenominatorShift-Down	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	FractionDenominator-DisplayStyleShiftDown	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: StackBottomDisplayStyleShiftDown.
46	FractionNumeratorGap-Min	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness
47	FractionNumDisplayStyle-GapMin	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
48	FractionRuleThickness	Thickness of the fraction bar. Suggested: default rule thickness.

\fontdimen	Dimension name	Description
49	FractionDenominatorGapMin	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	FractionDenomDisplayStyleGapMin	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: 3×default rule thickness.
51	SkewedFractionHorizontalGap	Horizontal distance between the top and bottom elements of a skewed fraction.
52	SkewedFractionVerticalGap	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	OverbarVerticalGap	Distance between the overbar and the (ink) top of he base. Suggested: 3×default rule thickness.
54	OverbarRuleThickness	Thickness of overbar. Suggested: default rule thickness.
55	OverbarExtraAscender	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	UnderbarVerticalGap	Distance between underbar and (ink) bottom of the base. Suggested: 3×default rule thickness.
57	UnderbarRuleThickness	Thickness of underbar. Suggested: default rule thickness.
58	UnderbarExtraDescender	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	RadicalVerticalGap	Space between the (ink) top of the expression and the bar over it. Suggested: 1¼ default rule thickness.
60	RadicalDisplayStyleVerticalGap	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + ¼ x-height.
61	RadicalRuleThickness	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	RadicalExtraAscender	Extra white space reserved above the radical. Suggested: RadicalRuleThickness.

\fontdimen	Dimension name	Description
63	RadicalKernBeforeDegree	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	RadicalKernAfterDegree	Negative kern after the degree of a radical, if such is present. Suggested: −10/18 of em.
65	RadicalDegreeBottom-RaisePercent	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.