

# Experimental Unicode mathematical typesetting: The unicode-math package

WILL ROBERTSON

*Philipp Stephani, Joseph Wright, Khaled Hosny, and others*

<http://github.com/wspr/unicode-math>

2019/09/26 0.8p

## Abstract

This document describes the unicode-math package, which is intended as an implementation of Unicode maths for  $\LaTeX$  using the  $X\TeX$  and  $\text{Lua}\TeX$  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  $X\TeX$  and  $\text{Lua}\TeX$ . 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

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Acknowledgements</b>	<b>4</b>
<b>3</b>	<b>Getting started</b>	<b>4</b>
3.1	New commands . . . . .	4
3.2	Package options . . . . .	6
<b>4</b>	<b>Unicode maths font setup</b>	<b>7</b>
4.1	Using multiple fonts . . . . .	7
4.1.1	Control over alphabet ranges . . . . .	8
4.2	Script and scriptscript fonts/features . . . . .	9
4.3	Maths ‘versions’ . . . . .	9
4.4	Legacy maths ‘alphabet’ commands . . . . .	10
4.4.1	Default ‘text math’ fonts . . . . .	10
4.4.2	Replacing ‘text math’ fonts by symbols . . . . .	10
4.4.3	Operator font . . . . .	11
<b>5</b>	<b>Maths input</b>	<b>11</b>
5.1	Math ‘style’ . . . . .	11
5.2	Bold style . . . . .	13
5.3	Sans serif style . . . . .	13
5.3.1	What about bold sans serif? . . . . .	14
5.4	All (the rest) of the mathematical styles . . . . .	14
5.4.1	Scope of the functionality of the <code>\sym..</code> commands . . . . .	14
5.4.2	Double-struck . . . . .	14
5.4.3	Caligraphic vs. Script variants . . . . .	15
5.5	Miscellanea . . . . .	16
5.5.1	Nabla . . . . .	16
5.5.2	Partial . . . . .	16
5.5.3	Primes . . . . .	16
5.5.4	Subscripts and superscripts and symbol alphabets . . . . .	17
5.5.5	Unicode subscripts and superscripts . . . . .	17
5.5.6	Colon . . . . .	17
5.5.7	Slashes and backslashes . . . . .	18
5.5.8	Behaviour of hyphens in mathematics . . . . .	19
5.5.9	Growing and non-growing accents . . . . .	20
5.5.10	Negations and the <code>\not</code> command . . . . .	20
5.5.11	Pre-drawn fraction characters . . . . .	20
5.5.12	Circles . . . . .	21
5.5.13	Triangles . . . . .	21

<b>6</b>	<b>Advanced</b>	<b>22</b>
6.1	Warning messages . . . . .	22
6.2	How to overwrite a macro . . . . .	22
6.3	Programmer's interface . . . . .	22
<b>A</b>	<b>stix table data extraction</b>	<b>23</b>
<b>B</b>	<b>Documenting maths support in the NFSS</b>	<b>23</b>
<b>C</b>	<b>Legacy T<sub>E</sub>X font dimensions</b>	<b>25</b>
<b>D</b>	<b>X<sub>Y</sub>T<sub>E</sub>X math font dimensions</b>	<b>25</b>

## 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. (`XYTeX`-only at time of writing.) Note that `unicode-math` and `mathspec` are not compatible with each other.

## 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 `XYTeX`; 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 `XYLATeX` and `LuaLATeX`)

```
\usepackage{amsmath}
\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 will be loaded automatically.

Note that `amsmath` should be loaded before `unicode-math` to avoid conflicts, and it will be loaded automatically if not requested earlier in the preamble.

### 3.1 New commands

`LATeX`, since the first version of `LATeX 2ε`, 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 some problems fitting into this design. 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\i}`.

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.

It was 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 with a command like `\mathbf` for the boldface text font in math and a command like `\symbf` (for the bold math font).

The five new symbol font commands that behave in this way are: `\symup`, `\symit`, `\symbf`, `\symsf`, and `\symtt`. These commands switch to single-letter mathematical symbols (generally within the same OpenType font).

The legacy `\math.` commands switch to text fonts that are 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.

In addition, `unicode-math` also provides a number of commands (such as `\symcal`) to select specific ‘symbol alphabets’ within the `unicode maths` font, with usage, e.g., `\symcal{G}`  $\rightarrow$   $\mathcal{G}$ . The full listing is shown in Table 1. For backwards compatibility, many of these are also defined with ‘familiar’ synonyms such as `\mathcal`. 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.

Using the `\sym.` commands, the glyphs used to produce PDF output are Unicode-encoded, and therefore a symbol such as  $\mathcal{G}$  can be copy-pasted into another programme (or even into the source of another `LATEX` document using `unicode-math`) without loss of meaning. This is an important aspect of Unicode mathematics, but the `unicode-math` package is not ‘pure’ in the sense that the package also provides other mechanisms to change the fonts used in the PDF output; the philosophy of the package is to provide mechanisms for technical authors

Table 1: New unicode-math commands which overlap with legacy math commands. For new documents the sym versions are recommended.

Command	Synonym	Command	Synonym
<code>\symnormal</code>	<code>\mathnormal</code>		
<code>\symliteral</code>			
		<code>\ymbfsf</code>	<code>\mathbfsf</code>
		<code>\ymbfup</code>	<code>\mathbfup</code>
		<code>\ymbfit</code>	<code>\mathbfit</code>
<code>\ymbb</code>	<code>\mathbb</code>		
<code>\ymbbit</code>	<code>\mathbbit</code>		
<code>\symcal</code>	<code>\mathcal</code>	<code>\ymbfcal</code>	<code>\mathbfcal</code>
<code>\symscr</code>	<code>\mathscr</code>	<code>\ymbfscr</code>	<code>\mathbfscr</code>
<code>\symfrak</code>	<code>\mathfrak</code>	<code>\ymbffrak</code>	<code>\mathbffrak</code>
<code>\symsfup</code>	<code>\mathsfup</code>	<code>\ymbfsfup</code>	<code>\mathbfsfup</code>
<code>\symffit</code>	<code>\mathffit</code>	<code>\ymbffsit</code>	<code>\mathbffit</code>

to invent and experiment with new syntaxes for their work.

### 3.2 Package options

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

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

Note, however, that some package options affect 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{texgyrepagella-math.otf}
% OR
\setmathfont{texgyrepagella-math.otf}[math-style=TeX]
```

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

Table 2: Package options.

Option	Description	See...
<code>math-style</code>	Style of letters	§5.1
<code>bold-style</code>	Style of bold letters	§5.2
<code>sans-style</code>	Style of sans serif letters	§5.3
<code>nabla</code>	Style of the nabla symbol	§5.5.1
<code>partial</code>	Style of the partial symbol	§5.5.2
<code>colon</code>	Behaviour of <code>\colon</code>	§5.5.6
<code>slash-delimiter</code>	Glyph to use for ‘stretchy’ slash	§5.5.7

Table 3: Maths font options.

Option	Description	See...
<code>range</code>	Style of letters	section §4.1
<code>script-font</code>	Font to use for sub- and super-scripts	section §4.2
<code>script-features</code>	Font features for sub- and super-scripts	section §4.2
<code>sscript-font</code>	Font to use for nested sub- and super-scripts	section §4.2
<code>sscript-features</code>	Font features for nested sub- and super-scripts	section §4.2

## 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 `srix` table) provides the mapping between Unicode maths glyphs and macro names (all 3298 — or however many — of them!). A single command

$$\setmathfont{\langle font name \rangle}[\langle font features \rangle]$$

implements this for every every symbol and alphabetic variant. That means  $x$  to  $x$ ,  $\xi$  to  $\xi$ ,  $\leq$  to  $\leq$ , etc.,  $\mathcal{H}$  to  $\mathcal{H}$  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.

### 4.1 Using multiple fonts

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

$$\setmathfont{\langle font name \rangle}[\text{range}=\langle unicode range \rangle, \langle font features \rangle]$$

where *<unicode range>* is a comma-separated list of Unicode slot numbers and ranges such as {"27D0-"27EB,"27FF,"295B-"297F}. Note that T<sub>E</sub>X's syntax for accessing the slot number of a character, such as `\+`, will also work here. Only numerical slots can be used in ranged declarations.

Note that, for efficiency, the `unicode-math` package only loads a default maths setup when absolutely necessary. Before you use the `range` option you must first load a 'main' maths font in the standard way.

You may also use the macro for accessing the glyph, such as `range=\int`, or whole collection of symbols with the same math type, such as `range=\mathopen`, or complete math styles such as `range=\symbb` (or just `range=bb`).

#### 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, bb, bbit, scr, cal, bfcalf, frak, tt, sfup,
sfit, bfup, bfit, bfscr, bffrak, bfsfup, bfsfit
```

Fonts can be selected (for predefined 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.

A common request is to load numerals only from a specific font. This can be achieved with an option such as `range=up/{num}`.

Note that 'meta-styles' such as 'bf' and 'sf' are not included in the list above 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{SomeFrakturFont}[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}
  }
]

```

## 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  $A_{BC}$ ). 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>}`.

Note there are currently open issues regarding the interaction between the version and the range features, so please proceed with caution.

#### 4.4 Legacy maths ‘alphabet’ commands

L<sup>A</sup>T<sub>E</sub>X 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. (These definitions hold regardless of package option, in case you need to be sure.)

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 L<sup>A</sup>T<sub>E</sub>X, `\mathsf` doesn’t change even if the main document font is changed using `\renewcommand\sfddefault{...}`. With `unicode-math` loaded, after writing `\setsansfont{Helvetica}`, `\mathsf` will now be set in Helvetica.

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.

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’
<code>\mathrm=text</code>	<code>\mathrm=sym</code>
<code>\mathup=text*</code>	<code>\mathup=sym*</code>
<code>\mathit=text</code>	<code>\mathit=sym</code>
<code>\mathsf=text</code>	<code>\mathsf=sym</code>
<code>\mathbf=text</code>	<code>\mathbf=sym</code>
<code>\mathtt=text</code>	<code>\mathtt=sym</code>

#### 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 ‘*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 five (case sensitive) arguments: `TeX`, `ISO`, `french`, `upright`, or `literal`.<sup>1</sup> The `math-style` options’ effects are shown in brief in table 5.

<sup>1</sup>Interface inspired by Walter Schmidt’s `lucimatx` package.

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

Package option	Example	
	Latin	Greek
<code>math-style=ISO</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=TeX</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=french</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$
<code>math-style=upright</code>	$(a, z, B, X)$	$(\alpha, \beta, \Gamma, \Xi)$

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.

For compatibility and consistency, however, upright and italic Greek letters can be 'forced' using `up` or `it` prefixes before their names. For example, `\Gamma` will give an upright or italic Gamma depending on the `math-style`, but `\upGamma` and `\itGamma` will always give upright or italic Gammas, respectively.

*'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.

*'Full-width' letters* Unicode contains 'full-width' versions of ASCII from `U+FF01`. The numerals and latin letters in this range are defined by `unicode-math` to map to their standard ASCII counterparts, which are then controlled by the relevant `math-style` setting. Other full-width symbols are not currently included but can be if there is sufficient need or desire.

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

Package option	Example	
	Latin	Greek
<code>bold-style=ISO</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$
<code>bold-style=TeX</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$
<code>bold-style=upright</code>	$(\mathbf{a}, \mathbf{z}, \mathbf{B}, \mathbf{X})$	$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Xi})$

## 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 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  $\boldsymbol{\zeta} = (\zeta_r, \zeta_\phi, \zeta_\theta)$ . Confusingly, the syntax in  $\LaTeX$  traditionally has been different for obtaining 'normal' bold symbols in Latin and Greek: `\mathbf` in the former (' $\mathbf{M}$ '), and `\bm` (or `\boldsymbol`, deprecated) in the latter (' $\boldsymbol{\zeta}$ ').

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 `\symsfup`, `\symsfit`, `\symbfsfup`, and `\symbfsfit` commands discussed in section §5.4.

How should the generic `\symsf` 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, the package options `[sans-style=upright]` and `[sans-style=italic]` control the behaviour of `\symsf`. 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 `\symsf` to either `\symsfup` or `\symsfit`, 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 (etc.). Therefore, bold sans serif follows from the setting for sans serif; it is completely independent of the setting for bold.

In other words, `\symsfsf` is either `\symsfsfup` or `\symsfsfit` based on `[sans-style=upright]` or `[sans-style=italic]`, respectively. And `[sans-style=literal]` causes `\symsfsf` 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  $\alpha'$ ), while `\symsfsf{\alpha}` gives  $\alpha'$  or  $\alpha'$  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 `\symsfsf{...}` rather than `\symsf{\symsf{...}}`. This may change in the future.

### 5.4.1 *Scope of the functionality of the \sym.. commands*

The `\sym..` commands are designed to affect only the follow sets of input letters: numerals (0–9), Latin ( $a-z, A-Z$ ), Greek ( $\alpha-\omega, A-\Omega$ ), and the `\partial` and `\nabla` symbols ( $\partial, \nabla$ ). These are the only symbols for which Unicode defines separate codepoints with varying mathematical style.

There is currently no scope for including other symbols in the `\sym..` commands, such as writing `\symsf{\int}` for a bold integral symbol. Therefore the commands provided by `unicode-math` should not be compared to those provided by the `bm` package.

### 5.4.2 *Double-struck*

The double-struck style (also known as 'blackboard bold') consists of upright Latin letters  $\{a-z, A-Z\}$ , numerals  $0-9$ , summation symbol  $\Sigma$ , and four Greek letters only:  $\{\gamma, \pi, \Gamma, \Pi\}$ .

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

Style	Font			Alphabet		
	Shape	Series	Switch	Latin	Greek	Numerals
Serif	Upright	Normal	<code>\symup</code>	•	•	•
		Bold	<code>\sybfup</code>	•	•	•
	Italic	Normal	<code>\symit</code>	•	•	◦
		Bold	<code>\sybfif</code>	•	•	◦
Sans serif	Upright	Normal	<code>\symsfup</code>	•		•
		Bold	<code>\sybfsfup</code>	•	•	•
	Italic	Normal	<code>\symsfit</code>	•		◦
		Bold	<code>\sybfsfit</code>	•	•	◦
Typewriter	Upright	Normal	<code>\symtt</code>	•		•
Double-struck	Upright	Normal	<code>\sybbb</code>	•		•
	Italic	Normal	<code>\sybbbit</code>	•		
Script	Upright	Normal	<code>\symscr</code>	•		
		Bold	<code>\sybfscr</code>	•		
Fraktur	Upright	Normal	<code>\symfrak</code>	•		
		Bold	<code>\sybffrac</code>	•		

While `\sybbb{\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: *Ddeijj*. 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.

#### 5.4.3 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 separate 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. An example is shown below.

The Script style (`\mathscr`) in XITS Math is: *ABCXYZ*

The Caligraphic style (`\mathcal`) in XITS Math is: *ABCXYZ*

Table 8: The various forms of nabla.

Description		Glyph
Upright	Serif	$\nabla$
	Bold serif	$\blacktriangledown$
	Bold sans	$\blacktriangledown$
Italic	Serif	$\nabla$
	Bold serif	$\blacktriangledown$
	Bold sans	$\blacktriangledown$

Table 9: The partial differential.

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

## 5.5 *Miscellanea*

### 5.5.1 *Nabla*

The symbol  $\nabla$  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).  $\text{\TeX}$  classically uses an upright nabla, and iso standards agree with this convention. The package options `nabla=upright` (default) and `nabla=italic` switch between the two choices, and `nabla=literal` respects the shape of the input character. `nabla=literal` is activated automatically after `math-style=literal`.

These settings are then inherited through `\symbf`; `\symit` and `\symup` can be used to force the shape of the nabla one way or the other.

### 5.5.2 *Partial*

The same logic as for nabla applies to the symbols `u+2202` partial differential and `u+1D715` math italic partial differential. However, in practice these symbols are often designed identically in an italic style.

If the font you are using supports it, use the `partial=upright` or `partial=italic` (default) 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. `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



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 the ‘DejaVu Serif’ font, 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.

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 }
\ExplSyntaxOff
```

Backwards or reverse primes behave in exactly the same way; use the ASCII backtick (‘) or the Unicode reverse prime  $\text{U+2035}$  (‘). 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 Subscripts and superscripts and symbol alphabets

In traditional  $\text{\LaTeX}$ , users have for many years exploited a loophole in the implementation of `\mathrm` and similar to write expressions such as `x_\mathrm f` to achieve  $x_f$  instead of writing the more correct `x_{\mathrm{f}}`. Shorthand notation such as `x_\mathrm f` is not officially documented  $\text{\LaTeX}$  syntax, and due to a particular implementation detail in `unicode-math` this *incorrect* syntax is no longer supported.

#### 5.5.5 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.

#### 5.5.6 Colon

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

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

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

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

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.

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.

### 5.5.7 Slashes and backslashes

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

In regular L<sup>A</sup>T<sub>E</sub>X 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  $\Sigma$ ).

*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:  $A \setminus B$ .<sup>2</sup> 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’:  $\pi \approx 7 \setminus 22$ .<sup>3</sup> The  $\LaTeX$  name for this character is `\setminus`.

Finally, U+29F9 big reverse solidus is a ‘big operator’ (like  $\Sigma$ ).

*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

$$\left[ \begin{array}{cc} a & b \\ c & d \end{array} \right] / \left[ \begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right] )$$

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.)

### 5.5.8 Behaviour of hyphens in mathematics

Unicode defines the following related characters:

- U+002D hyphen-minus

<sup>2</sup>§4.4.5.11 <http://www.w3.org/TR/MathML3/>

<sup>3</sup>This is valid syntax in the Octave and Matlab programming languages, in which it means matrix inverse pre-multiplication. I.e.,  $A \setminus B \equiv A^{-1}B$ .

- $\text{U+02212}$  minus sign
- $\text{U+02010}$  hyphen (`\mathhyphen`)

The first two of these characters in the input will all behave as the binary operator ‘minus sign’. The third is defined by `unicode-math` as a ‘math letter’ for constructions like  $\mathbb{R} \bmod \text{Mod}$  ( $R\text{-Mod}$ ). If more control is needed surrounding these symbols, additional options can be added to the package; please get in touch if this is the case for you.

### 5.5.9 Growing and non-growing accents

There are a few accents for which  $\text{\TeX}$  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  $\text{\XeTeX}$  and  $\text{\LuaTeX}$  did not support this distinction, however, and *all* accents there were growing automatically. (I.e., `\hat` and `\widehat` are equivalent.) As of  $\text{\LuaTeX}$  v0.65 and  $\text{\XeTeX}$  v0.9998, these wide/non-wide commands will again behave in their expected manner.

### 5.5.10 Negations and the `\not` command

The `\not` command in classic  $\text{\LaTeX}$  2<sub>ε</sub> was a mathematical slash modifying glyph that allowed for ‘negating’ maths symbols where pre-built glyphs were not available. While Unicode encodes a slot for this modifying slash, it is only well-supported in  $\text{\LuaTeX}$  and not in  $\text{\XeTeX}$ .

To provide more flexibility, the `unicode-math` package defines `\not` to search for a predefined ‘negated’ definitions for its argument and use that if available. This method can be used for fine-tuning in cases where spacing needs to be adjusted.

A ‘negated definition’ is any symbol command prefixed by either `n` or `not`. For example, `unicode-math` by default defines both `\leftarrow` ( $\leftarrow$ ) and `\nleftarrow` ( $\nleftarrow$ ).

To define custom negated definitions for either symbols (e.g., `\not=`) or commands (e.g., `\not\equal`), use the `\NewNotCommand{<symbol or cmd>}{<definition>}` command. Its usage is as follows:

```
\NewNegationCommand {=} {\neq}
\NewNegationCommand {\equal} {\neq}
```

If the command has already been defined, an error will result and `\RenewNegationCommand` can be used to overwrite the original definition.

### 5.5.11 Pre-drawn fraction characters

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

Slot	Command	Glyph	Glyph	Command	Slot
U+00B7	<code>\cdot</code>	·			
U+22C5	<code>\cdot</code>	·			
U+2219	<code>\vysmblkcircle</code>	•	◦	<code>\vysmwhtcircle</code>	U+2218
U+2022	<code>\smbllkcircle</code>	•	◦	<code>\smwhtcircle</code>	U+25E6
U+2981	<code>\mdsmbllkcircle</code>	●	○	<code>\mdsmwhtcircle</code>	U+26AC
U+26AB	<code>\mdblkcircle</code>	●	○	<code>\mdwhtcircle</code>	U+26AA
U+25CF	<code>\mdlglkcircle</code>	●	○	<code>\mdlgwhtcircle</code>	U+25CB
U+2B24	<code>\lgblkcircle</code>	●	○	<code>\lgwhtcircle</code>	U+25EF

Table 11: Filled and hollow Unicode circles.

$\frac{1}{4}$   $\frac{1}{2}$   $\frac{3}{4}$   $\frac{2}{3}$   $\frac{1}{7}$   $\frac{1}{9}$   $\frac{1}{10}$   $\frac{1}{3}$   $\frac{2}{3}$   $\frac{1}{5}$   $\frac{2}{5}$   $\frac{3}{5}$   $\frac{4}{5}$   $\frac{1}{6}$   $\frac{5}{6}$   $\frac{1}{8}$   $\frac{3}{8}$   $\frac{5}{8}$   $\frac{7}{8}$

For example, instead of writing `\tfrac{12}{x}`, you may consider it more readable to have `\frac{x}{2}` 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.

### 5.5.12 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.

L<sup>A</sup>T<sub>E</sub>X defines considerably fewer: `\circ` and `\bigcirc` for white; `\bullet` for black. This package maps those commands to `\vysmwhtcircle`, `\mdlgwhtcircle`, and `\smbllkcircle`, respectively.

### 5.5.13 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 T<sub>E</sub>X, 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.

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:  $\Delta x$ .

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

Slot	Command	Glyph	Class
U+25B5	<code>\vartriangle</code>	$\triangle$	binary
U+25B3	<code>\bigtriangleup</code>	$\bigtriangleup$	binary
U+25B3	<code>\triangle</code>	$\triangle$	ordinary
U+2206	<code>\increment</code>	$\Delta$	ordinary
U+0394	<code>\mathup\Delta</code>	$\Delta$	ordinary

Table 12: Different upwards pointing triangles.

## 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 turned 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 *How to overwrite a macro*

`unicode-math` defines the macros by `\AtBeginDocument`, namely delays the definition until `\begin{document}` is met. If you want to overwrite a macro defined by `unicode-math`, please redefine it in `\AtBeginDocument` after loading this package.

### 6.3 *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_t1`. It will contain the maths style without the leading ‘`math`’ string; for example, `\symbf { \show \l_@@_mathstyle_t1 }` 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](http://www.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,  $\langle NFSS\ decl. \rangle$  stands for something like  $\{\text{T1}\}\{\text{lmr}\}\{\text{m}\}\{\text{n}\}$ .

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

```
\DeclareSymbolFont{\name}\langle NFSS decl. \rangle
```

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}\langle NFSS decl. \rangle
```

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}\langle NFSS decl. \rangle
```

```
\SetMathAlphabet{\cmd}\{\maths version}\langle NFSS decl. \rangle
```

**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  $X_{\text{T}}\TeX$ . 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  $\X_{\text{L}}\text{T}_{\text{E}}\text{X}$ .

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	Maths font, $\backslash\text{fam2}$	Maths font, $\backslash\text{fam3}$
$\phi_1$ slant per pt	$\sigma_5$ x height	$\xi_8$ default rule thickness
$\phi_2$ interword space	$\sigma_6$ quad	$\xi_9$ big op spacing1
$\phi_3$ interword stretch	$\sigma_8$ num1	$\xi_{10}$ big op spacing2
$\phi_4$ interword shrink	$\sigma_9$ num2	$\xi_{11}$ big op spacing3
$\phi_5$ x-height	$\sigma_{10}$ num3	$\xi_{12}$ big op spacing4
$\phi_6$ quad width	$\sigma_{11}$ denom1	$\xi_{13}$ big op spacing5
$\phi_7$ extra space	$\sigma_{12}$ denom2	
$\phi_8$ cap height ( $\text{X}\TeX$ only)	$\sigma_{13}$ sup1	
	$\sigma_{14}$ sup2	
	$\sigma_{15}$ sup3	
	$\sigma_{16}$ sub1	
	$\sigma_{17}$ sub2	
	$\sigma_{18}$ sup drop	
	$\sigma_{19}$ sub drop	
	$\sigma_{20}$ delim1	
	$\sigma_{21}$ delim2	
	$\sigma_{22}$ axis height	

## D $\text{X}\TeX$ math font dimensions

These are the extended  $\backslash\text{fontdimens}$  available for suitable fonts in  $\text{X}\TeX$ . Note that  $\text{Lua}\TeX$  takes an alternative route, and this package will eventually provide a wrapper interface to the two (I hope).

$\backslash\text{fontdimen}$	Dimension name	Description
10	SCRIPTPERCENTSCALEDOWN	Percentage of scaling down for script level 1. Suggested value: 80%.
11	SCRIPTSCRIPTPERCENTSCALEDOWN	Percentage of scaling down for script level 2 (ScriptScript). Suggested value: 60%.
12	DELIMITEDSUBFORMULAMINHEIGHT	Minimum height required for a delimited expression to be treated as a subformula. Suggested value: normal line height $\times$ 1.5.
13	DISPLAYOPERATORMINHEIGHT	Minimum height of n-ary operators (such as integral and summation) for formulas in display mode.

<code>\fontdimen</code>	Dimension name	Description
14	<code>MATHLEADING</code>	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 ( <code>os2.sTypoAscender + os2.sTypoLineGap - MathLeading</code> ) or with ink going below <code>os2.sTypoDescender</code> will result in increasing line height.
15	<code>AXISHEIGHT</code>	Axis height of the font.
16	<code>ACCENTBASEHEIGHT</code>	Maximum (ink) height of accent base that does not require raising the accents. Suggested: x-height of the font ( <code>os2.sxHeight</code> ) plus any possible overshots.
17	<code>FLATTENEDACCENTBASEHEIGHT</code>	Maximum (ink) height of accent base that does not require flattening the accents. Suggested: cap height of the font ( <code>os2.sCapHeight</code> ).
18	<code>SUBSCRIPTSHIFTDOWN</code>	The standard shift down applied to subscript elements. Positive for moving in the downward direction. Suggested: <code>os2.ySubscriptYOffset</code> .
19	<code>SUBSCRIPTTOPMAX</code>	Maximum allowed height of the (ink) top of subscripts that does not require moving subscripts further down. Suggested: $\frac{1}{5}$ x-height.
20	<code>SUBSCRIPTBASELINEDROPMIN</code>	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	<code>SUPERSCRIPSHIFTUP</code>	Standard shift up applied to superscript elements. Suggested: <code>os2.ySuperscriptYOffset</code> .
22	<code>SUPERSCRIPSHIFTUPCRAMPED</code>	Standard shift of superscripts relative to the base, in cramped style.
23	<code>SUPERSCRIPBOTTOMMIN</code>	Minimum allowed height of the (ink) bottom of superscripts that does not require moving subscripts further up. Suggested: $\frac{1}{4}$ x-height.

<code>\fontdimen</code>	Dimension name	Description
24	<code>SUPERSCRIPBASELINEDROP- MAX</code>	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	<code>SUBSUPERSCRIPGAPMIN</code>	Minimum gap between the superscript and subscript ink. Suggested: 4×default rule thickness.
26	<code>SUPERSCRIPBOTTOMMAX- WITHSUBSCRIPT</code>	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: $\frac{1}{5}$ x-height.
27	<code>SPACEAFTERSCRIP</code>	Extra white space to be added after each subscript and superscript. Suggested: 0.5pt for a 12 pt font.
28	<code>UPPERLIMITGAPMIN</code>	Minimum gap between the (ink) bottom of the upper limit, and the (ink) top of the base operator.
29	<code>UPPERLIMITBASELINERISEMIN</code>	Minimum distance between baseline of upper limit and (ink) top of the base operator.
30	<code>LOWERLIMITGAPMIN</code>	Minimum gap between (ink) top of the lower limit, and (ink) bottom of the base operator.
31	<code>LOWERLIMITBASELINEDROP- MIN</code>	Minimum distance between baseline of the lower limit and (ink) bottom of the base operator.
32	<code>STACKTOPSHIFTUP</code>	Standard shift up applied to the top element of a stack.
33	<code>STACKTOPDISPLAYSTYLESHIFT- UP</code>	Standard shift up applied to the top element of a stack in display style.
34	<code>STACKBOTTOMSHIFTDOWN</code>	Standard shift down applied to the bottom element of a stack. Positive for moving in the downward direction.
35	<code>STACKBOTTOMDISPLAYSTYLE- SHIFTDOWN</code>	Standard shift down applied to the bottom element of a stack in display style. Positive for moving in the downward direction.
36	<code>STACKGAPMIN</code>	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.

<code>\fontdimen</code>	Dimension name	Description
37	<code>STACKDISPLAYSTYLEGAPMIN</code>	Minimum gap between (ink) bottom of the top element of a stack, and the (ink) top of the bottom element in display style. Suggested: $7 \times$ default rule thickness.
38	<code>STRETCHSTACKTOPSHIFTUP</code>	Standard shift up applied to the top element of the stretch stack.
39	<code>STRETCHSTACKBOTTOMSHIFT-DOWN</code>	Standard shift down applied to the bottom element of the stretch stack. Positive for moving in the downward direction.
40	<code>STRETCHSTACKGAPABOVEMIN</code>	Minimum gap between the ink of the stretched element, and the (ink) bottom of the element above. Suggested: <code>UpperLimitGapMin</code>
41	<code>STRETCHSTACKGAPBELOWMIN</code>	Minimum gap between the ink of the stretched element, and the (ink) top of the element below. Suggested: <code>LowerLimitGapMin</code> .
42	<code>FRACTIONNUMERATORSHIFTUP</code>	Standard shift up applied to the numerator.
43	<code>FRACTIONNUMERATOR-DISPLAYSTYLESHIFTUP</code>	Standard shift up applied to the numerator in display style. Suggested: <code>StackTopDisplayStyleShiftUp</code> .
44	<code>FRACTIONDENOMINATORSHIFT-DOWN</code>	Standard shift down applied to the denominator. Positive for moving in the downward direction.
45	<code>FRACTIONDENOMINATOR-DISPLAYSTYLESHIFTDOWN</code>	Standard shift down applied to the denominator in display style. Positive for moving in the downward direction. Suggested: <code>StackBottomDisplayStyleShiftDown</code> .
46	<code>FRACTIONNUMERATORGAP-MIN</code>	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar. Suggested: default rule thickness
47	<code>FRACTIONNUMDISPLAYSTYLE-GAPMIN</code>	Minimum tolerated gap between the (ink) bottom of the numerator and the ink of the fraction bar in display style. Suggested: $3 \times$ default rule thickness.
48	<code>FRACTIONRULETHICKNESS</code>	Thickness of the fraction bar. Suggested: default rule thickness.

<code>\fontdimen</code>	Dimension name	Description
49	<code>FRACTIONDENOMINATORGAP-MIN</code>	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar. Suggested: default rule thickness
50	<code>FRACTIONDENOMDISPLAY-STYLEGAPMIN</code>	Minimum tolerated gap between the (ink) top of the denominator and the ink of the fraction bar in display style. Suggested: $3 \times$ default rule thickness.
51	<code>SKEWEDFRACTION-HORIZONTALGAP</code>	Horizontal distance between the top and bottom elements of a skewed fraction.
52	<code>SKEWEDFRACTIONVERTICAL-GAP</code>	Vertical distance between the ink of the top and bottom elements of a skewed fraction.
53	<code>OVERBARVERTICALGAP</code>	Distance between the overbar and the (ink) top of the base. Suggested: $3 \times$ default rule thickness.
54	<code>OVERBARRULETHICKNESS</code>	Thickness of overbar. Suggested: default rule thickness.
55	<code>OVERBAREXTRAASCENDER</code>	Extra white space reserved above the overbar. Suggested: default rule thickness.
56	<code>UNDERBARVERTICALGAP</code>	Distance between underbar and (ink) bottom of the base. Suggested: $3 \times$ default rule thickness.
57	<code>UNDERBARRULETHICKNESS</code>	Thickness of underbar. Suggested: default rule thickness.
58	<code>UNDERBAREXTRADESCENDER</code>	Extra white space reserved below the underbar. Always positive. Suggested: default rule thickness.
59	<code>RADICALVERTICALGAP</code>	Space between the (ink) top of the expression and the bar over it. Suggested: $1\frac{1}{4}$ default rule thickness.
60	<code>RADICALDISPLAYSTYLE-VERTICALGAP</code>	Space between the (ink) top of the expression and the bar over it. Suggested: default rule thickness + $\frac{1}{4}$ x-height.
61	<code>RADICALRULETHICKNESS</code>	Thickness of the radical rule. This is the thickness of the rule in designed or constructed radical signs. Suggested: default rule thickness.
62	<code>RADICALEXTRAASCENDER</code>	Extra white space reserved above the radical. Suggested: <code>RadicalRuleThickness</code> .

<code>\fontdimen</code>	Dimension name	Description
63	<code>RADICALKERNBEFOREDEGREE</code>	Extra horizontal kern before the degree of a radical, if such is present. Suggested: 5/18 of em.
64	<code>RADICALKERNAFTERDEGREE</code>	Negative kern after the degree of a radical, if such is present. Suggested: -10/18 of em.
65	<code>RADICALDEGREEBOTTOM-RAISEPERCENT</code>	Height of the bottom of the radical degree, if such is present, in proportion to the ascender of the radical sign. Suggested: 60%.