

Package **mathfont** v. 3.0a User Guide

Conrad Kosowsky

February 2026

`kosowsky.latex@gmail.com`

For easy, off-the-shelf use, type the following in your preamble and compile with \XeTeX or \LuaTeX :

```
\usepackage[(font name)]{mathfont}
```

As of version 2.0, using \LuaTeX is recommended. Minor backwards incompatible changes in version 3.0.

Overview

The **mathfont** package adapts Unicode text fonts for math mode. The package allows the user to specify a default font for different classes of math symbols, and it enables Unicode input in math mode. The package provides tools to change the font locally for math alphabet characters. When typesetting with \LuaTeX , **mathfont** adds resizable delimiters, big operators, and a MathConstants table to text fonts.

Handling fonts in \TeX and \LaTeX is a notoriously difficult task because fonts are complicated.¹ The **mathfont** package addresses this situation by providing tools to use TrueType and OpenType fonts in math mode and in the main text, and this user guide explains how to operate **mathfont**. For version history and code implementation, see **mathfont-code.pdf**, and for a list of all symbols accessible with **mathfont**, see **mathfont-symbol-list.pdf**. Those two pdf files, this user guide, and four example files are included in the **mathfont** installation and are available on CTAN. Because Unicode text fonts are plentiful, I hope that this package expands the possibilities for typesetting math in \LaTeX .

1 Loading and Basic Functionality

Loading fonts for math typesetting is more complicated than for regular text. First, selecting

Acknowledgements: Thanks to Lyric Bingham for her work checking my Unicode hex values. Thanks to Matthew Braham, Sergio Callegari, Daniel Flipo, Nikos Platis, Shyam Sundar, Adrian Vollmer, Herbert Voss, and Andreas Zidak for pointing out bugs in previous versions of **mathfont**. Thanks to Jean-François Burnol for pointing out an error in the documentation in reference to his **mathastext** package.

¹The last few decades have seen huge advances in loading fonts with \TeX . Donald Knuth originally designed \TeX to load only fonts created with Metafont, and more recent engines such as Jonathan Kew’s \XeTeX and Hans Hagen, et al.’s \LuaTeX have extended \TeX ’s font-loading capabilities to Unicode. \XeTeX supports OpenType and TrueType fonts natively, and \LuaTeX can load OpenType fonts through the **luaotfload** package. Information on \XeTeX is available at <https://tug.org/xetex/>, and information on \LuaTeX is available at the official website for \LuaTeX : <http://www.luatex.org/>. See also Ulrike Fischer, et al., “**luaotfload**—OpenType ‘loader’ for Plain \TeX and \LaTeX ,” <https://ctan.org/pkg/luaotfload>. For discussion of fonts generally, see Yannis Haralambous, *Fonts & Encodings* (Sebastopol: O’Reilly Media, Inc., 2007).

Table 1: Comparison of General Font-Loading Packages

| | Text font | Math font |
|-------------------------------------|--|-----------------------|
| Traditional \TeX font | <code>mathastext</code> | No general package |
| Unicode font (for math typesetting) | <code>mathfont</code> or <code>mathspec</code> | (lua-)unicode-math |
| Unicode font (for text only) | <code>fontspec</code> | <code>fontspec</code> |

fonts for math mode, both in plain \TeX and in the NFSS, involves additional macros above and beyond what we need to load text fonts. Second, \TeX expects fonts for math to contain extra information for formatting equations.² Broadly speaking, we say that a *math font* contains this extra information, whereas a *text font* does not, and typesetting math with glyphs from one or more text fonts may yield messier equations than using a properly prepared math font. The functionality of `mathfont` then is twofold: (1) provide a wrapper around the NFSS commands for math typesetting that serves as a high-level interface; and (2) implement Lua \TeX callbacks that artificially convert text fonts into math fonts at loading.³ Although `mathfont` tries its best to get your fonts right, it may run into trouble when picking fonts to load. If this happens, you should declare your font family and shapes in the NFSS before setting any fonts with `mathfont`.

You must use one of \XeLaTeX or Lua \LaTeX to typeset a document with `mathfont`. You can load `mathfont` with the standard `\usepackage` syntax, and the package accepts five optional arguments. If you use Lua \TeX , the options `adjust` or `no-adjust` will manually specify whether `mathfont` should adapt text fonts for math mode, and `mathfont` selects `adjust` by default. If you use \XeTeX , `mathfont` cannot adjust any font objects with Lua callbacks, and either of these package options will cause an error.⁴ For this reason, using Lua \TeX with `mathfont` is recommended as of version 2.0. The options `default-loader` and `fontspec-loader` determine which font-loading code `mathfont` uses. If you load the package with the `default-loader` option, `mathfont` uses a built-in font-loader, and if you load the package with `fontspec-loader`, `mathfont` uses the font-loader from `fontspec`. If you load `mathfont` with any other optional argument, the package will interpret it as a font name and call `\documentfont` (described in the next section) on your argument. Doing so selects that font for the text of your document and for the character classes in the upper portion of Table 3.

The `mathfont` package is closely related to several other \LaTeX packages. The functionality is closest to that of `mathspec` by Andrew Gilbert Moschou.⁵ That package is compatible

²Specifically, this extra information is a set of large variants, math-specific parameter values associated with individual characters, and a `MathConstants` table. Also, math fonts often use slightly wider bounding boxes for letters in math mode—the Computer Modern *f* is a well-known example. (Compare math-mode \mathbb{f} and italic \mathbb{f} . Without italic correction, we have \mathbb{f} .) For this reason, `mathfont` provides an interface to enlarge the bounding boxes of most Unicode characters when they appear in math mode. See section 5 for details.

³Values for `MathConstants` table are different from but inspired by Ulrik Vieth, “Understanding the Aesthetics of Math Typesetting,” (Bach \TeX Conference, 2008) and Ulrik Vieth “OpenType Math Illuminated,” *TUGboat* 30 (2009): 22–31. See also Bogusław Jackowski, “Appendix G Illuminated,” *TUGboat* 27 (2006): 83–90.

⁴With \XeLaTeX , `mathfont` does not add big operators or resizable delimiters to the font. This means you will have to use the Computer Modern defaults, load a separate math font for resizable characters, or end up with a document where large operators and delimiters do not scale like they do normally.

⁵Andrew Gilbert Moschou, “`mathspec`—Specify arbitrary fonts for mathematics in \XeTeX ,” <https://ctan.org/pkg/mathspec>.

Table 2: Main User-Level Control Sequences Provided by `mathfont`

| Control Sequence | Use | Engine |
|--|---|------------------------|
| <code>\documentfont</code> | Set font for text and math | |
| <code>\mainfont</code> | Set font for text | |
| <code>\mathfont</code> | Set font for math | X _Y TeX and |
| <code>\mathfontshapes</code> | Declare font shapes in the preamble | LuaTeX |
| <code>\newmath<specifier></code> | Create local font-change command for math | |
| <code>\mathfontcommands</code> | Create several local font-change commands | |
| <code>\mathconstantsfont</code> | Select MathConstants table for use | |
| <code>\charmline</code> | Enlarge bounding box; change size of extensibles | |
| <code>\charmfile</code> | Call <code>\charmline</code> on each line from a file | |
| <code>\charminfo</code> | Print charm information of a character | LuaTeX only |
| <code>\charmtype</code> | Print type of a character for charm purposes | |
| <code>\hsurdfactor</code> | Move overline for square root horizontally | |
| <code>\vsurdfactor</code> | Move overline for square root vertically | |
| <code>\rulethicknessfactor</code> | Set rule thickness for square root, fractions | |

with X_YTeX only and loads text fonts for use in math. The `unicode-math` package is a main L^ATeX package for loading actual Unicode math fonts, and `lua-unicode-math` is a LuaTeX-specific adaptation.⁶ If you have a Unicode font with proper math support, rather than a text font that you want to use for equations, consider using those packages instead of `mathfont`. Users who want a text font for math with pdfL^ATeX should consider Jean-François Burnol’s `mathastext` because `mathfont` is incompatible with pdfTeX.⁷ Finally, you may be better off using `fontspec` if your document does not contain any math.⁸ The `fontspec` package is designed to load TrueType and OpenType fonts for text and provides a high-level interface for selecting OpenType font features. Table 1 summarizes this information.

2 Setting the Default Font

The `mathfont` package provides three commands for setting fonts in your document. The `\mathfont` command sets the font for various classes of characters when they appear in math mode, and after you call `\mathfont` on any class of characters, you can use Unicode input for that class of characters in math mode. The `\mainfont` command sets the font outside of equations. The control sequence `\documentfont` calls both `\mainfont` and `\mathfont`,

⁶Marcel Krueger, “lua-unicode-math—OpenType Math font support for LuaLaTeX,” <https://ctan.org/pkg/lua-unicode-math>; Will Robertson, et al., “unicode-math—Unicode mathematics support for XeTeX and LuaTeX,” <https://ctan.org/pkg/unicode-math>.

⁷Jean-François Burnol, “mathastext—Use the text font in maths mode,” <https://ctan.org/pkg/mathastext>. In several previous versions of this documentation, I mischaracterized the approach of `mathastext` to T_EX’s internal mathematics spacing. In fact, `mathastext` preserves and in some cases extends rules for space between various math-mode characters.

⁸Will Robertson and Khaled Hosny, “fontspec—Advanced font selection in X_YL^ATeX and LuaL^ATeX,” <https://ctan.org/pkg/fontspec>.

Table 3: Character Classes

| Keyword | Meaning | Default Shape Identifier | Alphabetic? |
|----------------------------|---------------------------------|--------------------------|------------------|
| <code>upper</code> | Upper-Case Latin | <code>italic</code> | Yes |
| <code>lower</code> | Lower-Case Latin | <code>italic</code> | Yes |
| <code>diacritics</code> | Diacritics | <code>upright</code> | Yes |
| <code>greekupper</code> | Upper-Case Greek | <code>upright</code> | Yes |
| <code>greeklower</code> | Lower-Case Greek | <code>italic</code> | Yes |
| <code>digits</code> | Arabic Numerals | <code>upright</code> | Yes |
| <code>operator</code> | Operator Font | <code>upright*</code> | N/A [†] |
| <code>delimiters</code> | Delimiter | <code>upright</code> | No |
| <code>radical</code> | Square Root Symbol | <code>upright</code> | No |
| <code>symbols</code> | Basic Math Symbols | <code>upright</code> | No |
| <code>bigops</code> | Big Operators | <code>upright</code> | No |
| <code>agreekupper</code> | Upper-Case Ancient Greek | <code>upright</code> | Yes |
| <code>agreeklower</code> | Lower-Case Ancient Greek | <code>italic</code> | Yes |
| <code>cyrillicupper</code> | Upper-Case Cyrillic | <code>upright</code> | Yes |
| <code>cyrilliclower</code> | Lower-Case Cyrillic | <code>italic</code> | Yes |
| <code>hebrew</code> | Hebrew | <code>upright</code> | Yes |
| <code>extsymbols</code> | Extended Math Symbols | <code>upright</code> | No |
| <code>arrows</code> | Arrows | <code>upright</code> | No |
| <code>extbigops</code> | Extended Big Operators | <code>upright</code> | No |
| <code>bb</code> | Blackboard Bold (double-struck) | <code>upright</code> | No |
| <code>cal</code> | Caligraphic | <code>upright</code> | No |
| <code>frak</code> | Fraktur | <code>upright</code> | No |
| <code>bcal</code> | Bold Caligraphic | <code>upright</code> | No |
| <code>bfrak</code> | Bold Fraktur | <code>upright</code> | No |

[†]Technically `opererator` is not a class of symbols but rather is a local font-change instruction similar to what is described in section 3 of this user guide.

calls `\mathconstantsfont` if the engine is Lua \TeX , and calls `\mathfontcommands` if in the document preamble. Each of these three commands accepts one optional argument and one mandatory argument, so the full syntax is

```

\documentfont[⟨rm, sf, tt, or empty⟩]{⟨font name⟩}
\mainfont[⟨rm, sf, tt, or empty⟩]{⟨font name⟩}
\mathfont[⟨keywords⟩]{⟨font name⟩}

```

More recent calls to any of these commands will override previous instructions, and all font changes through these macros are local. *This is a backwards-incompatible change in version 3.0.* For context, \LaTeX classifies fonts into family (what most users typically think of as a “font”), series (the weight, usually medium or bold), and shape (the appearance, usually upright or italic). The mandatory argument of these three commands specifies a font family, and the optional argument of `\mathfont` specifies font series and shape. To choose a series and shape outside of math mode, you should use the standard commands from the kernel.⁹

⁹Specifically, `\fontseries` and `\fontshape` locally select arbitrary series and shapes from the NFSS, and

Table 4: Shape Identifiers

| Identifier | Series | Uses NFSS Series | Shape | Uses NFSS Shape |
|---|--------|-------------------------------------|---------|------------------------------------|
| <code>upright</code> | medium | <code>\mddefault</code> | upright | <code>\shapedefault</code> |
| <code>italic</code> | medium | <code>\mddefault</code> | italic | <code>\itdefault</code> |
| <code>bold</code> | bold | <code>\bfdefault</code> | upright | <code>\shapedefault</code> |
| <code>bolditalic</code> | bold | <code>\bfdefault</code> | italic | <code>\itdefault</code> |
| <code>\langle series \rangle / \langle shape \rangle</code> | | <code>\langle series \rangle</code> | | <code>\langle shape \rangle</code> |

The optional argument for `\mainfont` and `\documentfont` determines how `mathfont` stores the font information. Using `rm`, `sf`, or `tt` will prompt `mathfont` to set both `\familydefault` and one of `\rmdefault`, `\sfdefault`, or `\ttdefault` respectively to the requested NFSS family. The corresponding `\rmfamily`, `\sffamily`, or `\ttfamily` command will switch to the font family thereafter. Providing an empty optional argument instructs `mathfont` to switch to the font family without saving the font name, and in this case, you will see no change in the effects of `\rmfamily` and friends. If you do not provide an optional argument, `mathfont` uses `rm`.

The optional argument for `\mathfont` should be a comma-separated list of keywords from Table 3, and it determines which math-mode characters will undergo a font change. When you use this command, `mathfont` sets the font family for every character in those keywords when in math mode to the mandatory argument. For a list of characters associated with the different keywords, see `mathfont_symbol_list.pdf`. If you do not specify an optional argument, `\mathfont` uses all keywords from the upper half of Table 3 (but not including `delimiters`, `radical`, or `bigops` characters in \TeX), so calling `\mathfont` with no optional argument is a fast way to change the font for most common math characters.

Characters in math mode often appear in different series or shapes from the same font family, and `mathfont` allows users to request particular series and shape combinations by using the shape identifiers in Table 4. By default, the characters in a `\langle keyword \rangle` will follow the shape identifier stored in `\langle keyword \rangle default`, so for example, `\upperdefault` is initially defined to be the character string `italic`. Table 3 contains the initial value of each default setting. To change the series and shape, you can redefine any `\langle keyword \rangle default` macro to a different shape identifier, or you can request a different shape identifier in the optional argument of `\mathfont`. If you say `=\langle shape identifier \rangle` immediately after a keyword and before the following comma, the package will apply your requested `\langle shape identifier \rangle` to that keyword. For example, if you say

```
\mathfont[upper=upright, lower=upright]{Baskerville}
```

you will see Latin letters appear in upright Baskerville in your equations.

You may include `*` character immediately after the shape identifier if you want `mathfont` to use unmodified characters from the font. This option is most appropriate for situations such as math operators where you want to use regular text in your equations and do not need enlarged bounding boxes. For example, the command

```
\mathfont[upper, lower, operator=italic*]{Times New Roman}
```

the kernel provides macros that reformat their argument such as `\textbf` and `\textit`.

changes all Latin letters to italic Times New Roman and switches math operators to the same italic text that you would see if you use Times New Roman outside of math. This feature does not do anything in \TeX .

The mandatory argument of all three font-setting commands should be a system font name or an NFSS font family name. If you use a system font, you can select OpenType features by putting a colon after the font name and following it with the appropriate tags.¹⁰ The syntax for specifying features depends on the font-loader: with the built-in font-loader, you should use the standard “+ $\langle tag \rangle$ ” or “- $\langle tag \rangle$ ” syntax, and when using `fontspec` as the font-loader, you should use that package’s high-level interface, as described in the `fontspec` documentation. For example, suppose you want math with oldstyle numbers. With the built-in font-loader, you should add “+onum,” or if using `fontspec`, you should add `Numbers=OldStyle` to your `\mathfont` command. So to load Adobe Garamond Pro with oldstyle numbering, you would say

```
\mathfont{Adobe Garamond Pro:+onum}
```

with the built-in font-loader or

```
\mathfont{Adobe Garamond Pro: Numbers=OldStyle}
```

if using `fontspec`. With the built-in font-loader, you should separate OpenType tags with semi-colons, and for `fontspec`, you should use commas.

When the mandatory argument of a font-changing command is an NFSS family name, `mathfont` uses that font family. When the engine is \TeX , you don’t have to do anything special and can ignore the rest of this paragraph. When the engine is \LuaTeX , `mathfont` expects the NFSS to contain a second entry for the same font family that uses base mode for loading.¹¹ If $\langle name \rangle$ is the mandatory argument, you should tell `mathfont` the family name for the base-mode version by storing it in the control sequence $\langle name \rangle$ -base. If you do not do this, you will see an error message. My intention is that the first font family is for text, and the base-mode font family is for math—using two different loading options provides the greatest access to OpenType font features throughout your document.

You can access blackboard-bold, calligraphic, or fraktur letters in three ways. First, the Unicode standard contains encoding slots for these types of letters, and the last five keywords in Table 3 access this portion of the Unicode table.¹² If you call `\mathfont` on one of these $\langle keyword \rangle$ s, the package defines the macro

```
\math\langle keyword \rangle\{ \langle text \rangle \}
```

which behaves like a local font-change command from the next section and converts Latin

¹⁰By default, `mathfont` enables standard ligatures, traditional \TeX ligatures, and lining numbers. The package sets `smcp` to `true` or `false` depending on whether it is attempting to load a small-caps font. For the full list of OpenType features, see <https://docs.microsoft.com/en-us/typography/opentype/spec/featurelist>.

¹¹The `luaotfload` package supports three modes for loading fonts: `node`, `base`, and `harf`. The default setting is `node` mode. It supports full OpenType features in text but none for math, whereas `base` mode supports fewer OpenType features, but they work in both text and math. Loading a font with `harf` mode engages HarfBuzz, a font shaper that arranges characters in potentially complicated layouts but also provides no access to OpenType features in math.

¹²The Math Alphanumeric Symbols block is U+1D400–U+1D7FF. Most blackboard-bold, calligraphic, and fraktur letters live in this portion of the Unicode table, although a few live in other places.

Table 5: Macros to Create Local Font-Change Commands with Preset Shapes

| Command | Series | Uses NFSS Series | Shape | Uses NFSS Shape |
|---|--------|-------------------------|-------------------|-----------------------------------|
| <code>\newmathrm</code> | Medium | <code>\mddefault</code> | Upright | <code>\shapedefault</code> |
| <code>\newmathit</code> | Medium | <code>\mddefault</code> | Italic | <code>\itdefault</code> |
| <code>\newmathbf</code> | Bold | <code>\bfdefault</code> | Upright | <code>\shapedefault</code> |
| <code>\newmathbf</code> <code>it</code> | Bold | <code>\bfdefault</code> | Italic | <code>\itdefault</code> |
| <code>\newmathsc</code> | Medium | <code>\mddefault</code> | Small Caps | <code>\scdefault</code> |
| <code>\newmathscit</code> | Medium | <code>\mddefault</code> | Italic Small Caps | <code>\scdefault\itdefault</code> |
| <code>\newmathbfsc</code> | Bold | <code>\bfdefault</code> | Small Caps | <code>\scdefault</code> |
| <code>\newmathbfscit</code> | Bold | <code>\bfdefault</code> | Italic Small Caps | <code>\scdefault\itdefault</code> |

letters into $\langle keyword \rangle$ style. For example,

```
\mathfont[bb]{STIXGeneral}
```

defines `\mathbbb` to typeset blackboard-bold letters using the glyphs from STIXGeneral. Second, you may have a Unicode font where the normal Latin letters are themselves double struck, calligraphic, or fraktur, and in that case, you should declare a local font-change command using the tools in the next section. Third, `mathfont` should play nicely with (local) font-changing macros from other packages, so you should have no problem using, for example, `amssymb` with `mathfont`.¹³

3 Local Font Changes

With `mathfont`, it is possible to create commands that locally change the font for math alphabet characters, i.e. those marked as alphabetic in Table 3. (I am slightly abusing notation. In this section, I mean local as in acting on a small portion of the document, not local with respect to grouping. All font-change commands from `mathfont` are local in that regard.) The commands from this section should appear only in the preamble. The eight commands in Table 5 accept a $\langle control sequence \rangle$ as their first mandatory argument and a $\langle font name \rangle$ as the second, so for example, the macro `\newmathrm` looks like

```
\newmathrm{\langle control sequence \rangle}{\langle font name \rangle}
```

You can specify OpenType features as part of the $\langle font name \rangle$ the same way as in the previous section. The macros in Table 5 define the control sequence in their first argument to accept a string of characters and convert it to the font family in the second argument. The series and shape information depend on the command used. For example, writing

```
\newmathrm{\matharial}{Arial}
```

creates the macro

```
\matharial{\langle argument \rangle}
```

which can be used only in math mode and which converts the math alphabet characters in

¹³American Mathematical Society, “`amssymb`.” See also American Mathematical Society, “`amsfonts`— \TeX fonts from the American Mathematical Society,” <https://ctan.org/pkg/amsfonts>.

its $\langle argument \rangle$ into the Arial font with upright shape and medium weight.

The macro `\mathfontcommands` accepts a single font name as its argument. It calls each `\newmath<specifier>` command from Table 5 to define `\math<specifier>` using the font family from its argument. For example,

```
\mathfontcommands{Helvetica}
```

makes the standard font-changing commands such as `\mathrm` and `\mathit` use Helvetica instead of Computer Modern.

In traditional L^AT_EX, `\mathrm` and friends do not change the font for Greek characters. That remains the case until you call `\mathfont` for `greeklower` or `greekupper` characters, and then any local font-change commands from this section will work on lower-case or upper-case Greek letters respectively. Similarly, the local font-change commands will affect Cyrillic and Hebrew characters after you call `\mathfont` for those keywords.

These nine control sequences provide tools for most local font changes, but they won't be able to address everything. Accordingly, `mathfont` provides general `\newmathfontcommand` macro. Its argument structure is

```
\newmathfontcommand{<control sequence>}{<font name>}{<series>}{<shape>}
```

where the $\langle control\ sequence \rangle$ in the first argument again becomes the macro that changes characters to the $\langle font\ name \rangle$. You are welcome to use a system font name when you call `\newmathfontcommand`, but my intention behind this command is that you can use an NFSS family name for the $\langle font\ name \rangle$. Then the series and shape values can correspond to more obscure font faces from the NFSS family that you would be otherwise unable to access. As with `\mathfont`, if you are using Lua_T_EX and the $\langle font\ name \rangle$ is an NFSS family, you should make sure that you defined `\langle font name \rangle-base`, or you will see an error.

4 Default Math Parameters

Lua_T_EX uses the `MathConstants` table from the most recent font assigned for use in math mode, and this means that in a document with multiple math fonts, the choice of `MathConstants` table can depend on the order of font declaration and be unpredictable. To avoid potential problems from using the wrong `MathConstants` table, `mathfont` provides

```
\mathconstantsfont[<shape identifier>]{<font name>}
```

where $\langle shape\ identifier \rangle$ is a shape identifier from Table 4 and $\langle font\ name \rangle$ is a font name as in the previous two sections. By default, `\mathconstantsfont` uses `upright` for the optional argument. This macro forces Lua_T_EX to use the `MathConstants` table from the requested font face, and once again, if you use an NFSS family name for $\langle font\ name \rangle$ with Lua_T_EX, make sure to define `\langle font name \rangle-base` appropriately. You don't need to set the `MathConstants` table when you use `\documentfont` or pass a font name to `mathfont` at loading because the package calls `\mathconstantsfont` automatically in this case. This command does not do anything in X_Y_T_EX.

5 Declaring Font Shapes

You are welcome to use the commands from sections 2 and 4 after `\begin{document}`, but

you are limited in what fonts you can use. \LaTeX prohibits new font declarations after the preamble, and at that point, `mathfont` will only use font faces that already exist in the NFSS or that it used for any purpose previously. What this means in practical terms: if you want to use `\mathfont` or `\mainfont` after `\begin{document}`, you probably have to use a font that `mathfont` handled in some capacity in the preamble. This makes it nonobvious how to use a font after the preamble if you do not want to set that font for any characters in the preamble.

To address this situation, `mathfont` provides the command `\mathfontshapes`. The purpose of this macro is to declare font shapes before `\begin{document}` without using them for anything. It accepts one optional argument and one mandatory argument, so the syntax is

```
\mathfontshapes[<shape identifiers>]{<font name>}
```

The optional argument should be a comma-separated list of shape identifiers, and by default, the macro uses `upright`, `upright*`, `italic`, and `bold`. The mandatory argument should be a font name like in the previous sections.

After you use `\mathfontshapes`, you should have no problem using the same font family for `\mainfont`. You will also be able to use the font family for `\mathfont` or `\mathconstantsfont` provided that you stick to shape identifiers that appeared in the optional argument of `\mathfontshapes`. For example, if you say

```
\mathfontshapes{Didot}
```

in your preamble, you can include

```
\mathfont{Didot}
```

after the preamble (even if you didn't have it in the preamble), but if you type

```
\mathfont[operator=italic*]{Didot}
```

you will run into problems because `\mathfontshapes` does not normally use the shape identifier `italic*`. You should instead say

```
\mathfontshapes[italic*]{Didot}
```

in your preamble to avoid any issues.

6 Lua Font Adjustments

When \LuaTeX reads in a font file, `mathfont` makes the font object in memory appear to be a math font rather than a text font. Most of this process happens behind the scenes, but `mathfont` provides control sequences that allow you to customize some modifications. With the exception of the read-only control sequences `\charminfo` and `\charmtype`, their wrapped equivalents, and the count variables, all macros from this section should appear in the preamble only.

The main macro for doing so is `\charmline`. (Charm stands for “character metric.”) It accepts one mandatory argument, so the syntax is

```
\charmline{<charm information>}
```

The *<charm information>* should be a sequence of entries separated by a comma and/or

Table 6: Number of Entries Required in `\charmline`

| Character | Type | Total Number of Entries |
|---|------|-------------------------|
| Everything Else | a | 5 |
| Delimiters, Radical (Surd Character), Big Operators | e | 34 |

space, where entry means one or two numbers or asterisks. If an entry is two numbers, two asterisks, or one of each, you must separate them with a slash, and the first entry in $\langle \textit{charm information} \rangle$ should be a single integer between 0 and 1,114,111. The command fully expands its argument, and you may see strange errors if $\langle \textit{charm information} \rangle$ contains unexpandable control sequences. Processing uses `tonumber()` from Lua, so your input can include anything that Lua recognizes as a number. In particular, you can specify hexadecimal numbers by prefixing them with `0x`.

This macro controls how **mathfont** adjusts bounding boxes, accent placement, and large variants. The first entry in $\langle \textit{charm information} \rangle$ corresponds to an entry in the Unicode table, and **mathfont** references the remaining entries when it alters properties of the character in that encoding slot during font loading. For purposes of charm information, a character can have one of two types: **a** for “alphabetic” or **e** for “extensible.”¹⁴ Any character that resizes, such as delimiters, big operators, and the surd character has type **e**, and you should specify 33 additional entries after the encoding slot. All other characters have type **a**, and you should specify 4 additional entries after the encoding slot. If you specify too few charm values, **mathfont** will raise an error, and if you provide too many, **mathfont** ignores the extras and prints a warning on the terminal. Version 3.0 eliminates type **u** and requires an extra entry for type **e** characters, and *these are backwards-incompatible changes*. If you call `\charmline` for an encoding slot that does not have any charm information, **mathfont** assumes that character should be type **a**. You can change a character to type **e** by including an `!` symbol immediately after the first entry, and you can change a character to type **a** by including a `?` symbol immediately after the first entry.

When **mathfont** parses $\langle \textit{charm information} \rangle$, the order of the entries determines their purpose. As is standard in $\text{T}_{\text{E}}\text{X}$, **mathfont** divides all numbers in `\charmline` by 1000 to form floats before using them in calculations. For type **a** characters, the second and third entries tell Lua $\text{T}_{\text{E}}\text{X}$ how much to stretch the left and right sides of the glyph’s bounding box when it appears in math mode, and the third entry determines horizontal placement of top math accents. The final entry determines the `bot_accent` attribute for that character in the font object, although Lua $\text{T}_{\text{E}}\text{X}$ does not use this piece of information for formatting equations.¹⁵ If the Unicode value corresponds to a type **e** character, **mathfont** interprets the first 30 entries after the encoding slot as 15 successive pairs of horizontal and vertical scale factors that determine the size of large variants. For example, the second and third entries in $\langle \textit{charm information} \rangle$ are the horizontal and vertical scale factors for the first large variant. The last three entries determine, respectively, horizontal placement of top accents,

¹⁴Alphabetic is something of a misnomer that I kept for historical reasons.

¹⁵See Hans Hagen and Mikael P. Sundqvist, “On Bottom Accents in OpenType Math,” *TUGBoat* 44 (2023): 207–208. See also two threads from the Lua $\text{T}_{\text{E}}\text{X}$ developer mailing list from 2012, available at <http://mailman.ntg.nl/archives/list/dev-luatex@ntg.nl/thread/3SDPLVHQYLVHOBUA3DWR0TWLMUBUQI5/#WX432F6M52MUIK4WE5I3W7YSJOWDIA4G> and <https://mailman.ntg.nl/archives/list/dev-luatex@ntg.nl/thread/PU2VYNG7XBWVBBUY4VIPJOS75NNZM7UE/#PU2VYNG7XBWVBBUY4VIPJOS75NNZM7UE>.

the `bot_accent` attribute, and the character’s italic correction.

Asterisks and subentries affect how `mathfont` uses charm information in memory. Writing an asterisk tells `mathfont` to use whatever value it has saved in memory, either the default value or the value from the most recent call to `\charmline` or `\charmfile`. When an entry contains two subentries, `mathfont` uses the first subentry for unslanted fonts and the second subentry for slanted or italic fonts. If an entry is a single number or asterisk, `mathfont` uses that input for both slanted and unslanted fonts.

If you want to store charm information in a file instead of your document preamble, you should use the control sequence `\charmfile`. This command accepts one mandatory argument, so the syntax is

```
\charmfile{<file name>}
```

It reads in a file and calls `\charmline` individually on each line from the file.

The default settings are decent, so for most applications, you can probably ignore charm information altogether. However, if you find bounding boxes or accent placement to be off slightly in your equations or if you want to change the scaling for a delimiter or big operator, you should try calling `\charmline` with different values to see what works. For a given character, the scale for changes is the width of that character’s unmodified bounding box.¹⁶ For example,

```
\charmline{0x61, 200, -100, *, 50}
```

tells `mathfont` to take the lower-case “a” (U+61, i.e. encoding slot number 97), increase the bounding box on the left side by 20% of the character width, decrease the bounding box on the right side by 10% of the character width, do nothing to the top accent, and shift the `bot_accent` attribute right by 5% of the character width. There is no general formula for what charm values to use for a given font! Rather, you will need to make a design choice based on what looks best, and if you regularly use a specific font, consider making a custom set of charm values and uploading it to CTAN.

The package provides two commands to query charm information in memory, namely `\charminfo` and `\charmttype`. These macros are weird in that they don’t take an argument. Rather, they scan (and remove) the next integer from the input stream in the same manner as a `\count` variable, so the syntax is

```
\charminfo<integer>
\charmttype<integer>
```

Both macros are fully expandable and work outside the document preamble. If `mathfont` has charm information in memory for the Unicode encoding slot `<integer>`, `\charmttype` expands to `a` or `e` depending on the type of that character, and `\charminfo` expands to a space-separated lists of charm entries. The results of `\charminfo` look exactly like the input of `\charmline`, so typing

```
\charmline{<encoding slot>},\charminfo{<encoding slot>}
```

is valid syntax and reassigns the charm information for `<encoding slot>` to itself. If `<integer>` is not the encoding slot for a character with charm information, both macros expand to `none`.

Three count variables affect equation formatting with `mathfont`. Their default value is 1000, and `mathfont` divides their value by 1000 to convert them to floats before using them.

¹⁶For type `a`, it’s technically the unmodified bounding box width plus italic correction because `mathfont` incorporates the italic correction into the character width.

Table 7: Lua Callbacks Created by **mathfont**

| Callback Name | What It Does By Default | Called? |
|----------------------------------|--|---|
| "mathfont.inspect_font" | Nothing | Always |
| "mathfont.pre_adjust" | Nothing | |
| "mathfont.disable_nomath" | Set font attribute <code>nomath</code> to false | |
| "mathfont.add_math_constants" | Create a <code>MathConstants</code> table | If base mode and <code>nomath</code> |
| "mathfont.fix_character_metrics" | Adjust bounding boxes, add character-specific math fields, create large variants | |
| "mathfont.post_adjust" | Nothing | |
| "mathfont.finishing_touces" | Nothing | Always |

The first two, `\hsurdfactor` and `\vsurdvactor`, affect the positioning of the overline in square root expressions. Specifically, changing either count scales the bounding box for the surd (square root) character by the corresponding float value. Larger values of `\hsurdvactor` therefore shift the overline to the right, and smaller values do the opposite. Larger values of `\vsurdfactor` raise the overline. The third count, `\rulethicknessfactor`, determines the thickness of horizontal rules such as the overline in square root expressions or the fraction bar. You will probably want to increase `\rulethicknessfactor` for fonts with heavier weight and decrease it for fonts with lighter weight. Please be aware that **mathfont** uses the value of each count only once per font when **TeX** loads the font for math typesetting. This typically happens the first time the user enters math mode.

The package provides user-friendly versions of each control sequence in this section. The macros `\CharmInfo` and `\CharmType` accept an integer as a single mandatory argument and print the results of `\charminfo` and `\charmtype` to the terminal. The macros `\SurdHorizontalFactor`, `\SurdVerticalFactor`, and `\RuleThicknessFactor` similarly accept a single integer as a mandatory argument and set the corresponding count variable to that value. Additionally, `\IntegralItalicFactor` accepts an integer as its argument and divides the argument by 1000 to form a float. When **TeX** loads a font, it increases the italic correction of the integral symbol by that fraction of the symbol's width, which affects positioning of limits. For parity in macro names and backwards compatibility, `\CharmLine` and `\CharmFile` are defined to do the same thing as their lower-case counterparts.

Advanced users who want to interact with the font adjustment process directly should use the seven callbacks in Table 7. When **luaotfload** loads a font, **mathfont** (1) always calls `mathfont.inspect_font`; (2) calls the middle five callbacks in the order that they appear in Table 7 if the font object contains `nomath=true` and was loaded with **base** mode; and (3) always calls `mathfont.finishing_touces`. Functions added to these callbacks should accept a font object as a single argument and return nothing. Further, please be careful when loading functions in `disable_nomath`, `add_math_constants`, and `fix_character_metrics`. If you add a function there, **LuaTeX** will not carry out the default behavior associated with the callback, so do not mess with these three callbacks unless you are duplicating the default behavior or you really know what you're doing. Otherwise, you risk breaking the package. See `mathfont-code.pdf` for more information.