Package mathfont v. 2.4 User Guide

Conrad Kosowsky
April 2025
kosowsky.latex@gmail.com

For easy, off-the-shelf use, type the following in your preamble and compile with XeLaTeX or LuaLaTeX:

\usepackage[⟨*font name*⟩]{mathfont}

As of version 2.0, using LuaLaTeX is recommended.

Overview

The mathfont package adapts unicode text fonts for math mode. The package allows the user to specify a default unicode font for different classes of math symbols, and it 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 load TrueType and OpenType fonts for use in math mode, and this document 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, particularly without built-in math support, 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 fonts for math mode, both in plain TeX and in the NFSS, involves additional macros above

Acknowledgements: Thanks to Lyric Bingham for her work checking my unicode hex values. Thanks to Daniel Flipo, 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.

[1] The last few decades have seen huge advances in loading fonts with TeX. Donald Knuth originally designed TeX to load fonts created with Metafont, and only 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	mathastext	No general package
Unicode font (for math typesetting)	mathfont or mathspec	unicode-math
Unicode font (for text only)	fontspec	fontspec

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 usually results in 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 LuaTEX 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 LuaLATEX 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 LuaTEX, 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 LuaTEX 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 `\setfont` (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 section of Table 2.

The mathfont package is closely related to several other LATEX packages. The functionality is closest to that of mathspec by Andrew Gilbert Moschou, which is compatible with XETEX only and selects characters from text fonts for math.[5] The unicode-math package is a main

[2]Specifically, 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 f and italic *f*.) For this reason, mathfont also provides an interface to enlarge the bounding boxes of Latin letters when they appear in math mode. See section 5 for details.

[3]Values for MathConstants table are different from but inspired by Ulrik Vieth, "Understanding the Æsthetics of Math Typesetting," (BachoTEX 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.

[4]With XELATEX, mathfont does not add big operators or resizable delimiters. 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.

[5]Andrew Gilbert Moschou, "mathspec—Specify arbitrary fonts for mathematics in XETEX," https://

Table 2: Character Classes

Keyword	Meaning	Default Shape	Alphabetic?
`upper`	Upper-Case Latin	Italic	Yes
`lower`	Lower-Case Latin	Italic	Yes
`diacritics`	Diacritics	Upright	Yes
`greekupper`	Upper-Case Greek	Upright	Yes
`greeklower`	Lower-Case Greek	Italic	Yes
`digits`	Arabic Numerals	Upright	Yes
`operator`	Operator Font	Upright	Yes
`delimiters`	Delimiter	Upright	No
`radical`	Square Root Symbol	Upright	No
`symbols`	Basic Math Symbols	Upright	No
`bigops`	Big Operators	Upright	No
`agreekupper`	Upper-Case Ancient Greek	Upright	Yes
`agreeklower`	Lower-Case Ancient Greek	Italic	Yes
`cyrillicupper`	Upper-Case Cyrillic	Upright	Yes
`cyrilliclower`	Lower-Case Cyrillic	Italic	Yes
`hebrew`	Hebrew	Upright	Yes
`extsymbols`	Extended Math Symbols	Upright	No
`arrows`	Arrows	Upright	No
`extbigops`	Extended Big Operators	Upright	No
`bb`	Blackboard Bold (double-struck)	Upright	No
`cal`	Caligraphic	Upright	No
`frak`	Fraktur	Upright	No
`bcal`	Bold Caligraphic	Upright	No
`bfrak`	Bold Fraktur	Upright	No

LaTeX package for loading actual unicode math fonts, and if you have a unicode font with proper math support, rather than a text font that you want to use for equations, consider using that package instead of mathfont.[6] Users who want a text font for math with pdfLaTeX 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. At the time of writing this document, I am not aware of a general font-loading

ctan.org/pkg/mathspec.

[6]Will Robertson, et al., "unicode-math—Unicode mathematics support for XeTeX and LuaTeX," https://ctan.org/pkg/unicode-math.

[7]Jean-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 TeX's internal mathematics spacing. In fact, mathastext preserves and in some cases extends rules for space between various math-mode characters.

[8]Will Robertson and Khaled Hosny, "fontspec—Advanced font selection in XeLaTeX and LuaLaTeX," https://ctan.org/pkg/fontspec.

Table 3: Commands Defined by `\setfont`

Command	Series	Shape
`\mathrm`	Medium	Upright
`\mathit`	Medium	Italic
`\mathbf`	Bold	Upright
`\mathbfit`	Bold	Italic
`\mathsc`	Medium	Small Caps
`\mathscit`	Medium	Italic Small Caps
`\mathbfsc`	Bold	Small Caps
`\mathbfscit`	Bold	Italic Small Caps

package for traditional TEX math fonts.

2 Setting the Default Font

The `\mathfont` command sets the default font for certain classes of characters when they appear in math mode. It accepts a single mandatory argument, which should be a system font name or a family name already present in the NFSS. The macro also accepts an optional argument, which should be a comma-separated list of keywords from Table 2, so the full syntax is

`\mathfont[`⟨*keywords*⟩`]{`⟨*font name*⟩`}`

When you use this command, mathfont sets the default font face for every character in those keywords to an upright or italic version of the font from the mandatory argument. See `mathfont_symbol_list.pdf` for a list of symbols corresponding to each keyword. If you do not include an optional argument, `\mathfont` acts on all keywords in the upper section of Table 2 (but not including `delimiters`, `radical`, or `bigops` characters in XETEX), so calling `\mathfont` with no optional argument is a fast way to change the font for most common math characters. To change the shape, you should say "`=upright`" or "`=italic`" immediately after the keyword and before the following comma, and spaces are allowed throughout the optional argument. For example, the command

```
\mathfont[lower=upright, upper=upright]{Times New Roman}
```

changes all Latin letters to upright Times New Roman. Once mathfont has set the default font for a keyword in Table 2, it will ignore any future instructions to change the font again for that keyword, and instead the package prints a warning message.

If you want to change the font for both text and math, you should use `\setfont` instead of `\mathfont`. This command accepts a single mandatory argument:

`\setfont{`⟨*font name*⟩`}`.

It calls `\mathfont` without an optional argument—i.e. for the default keywords—on your ⟨*font name*⟩ and sets your document's default text font to be the ⟨*font name*⟩. The command also defines the eight commands in Table 3 using the ⟨*font name*⟩ and the `\new` macros in the next section. Both `\mathfont` and `\setfont` should appear in the preamble only.

To select OpenType features, you should put a colon after the font name and follow it with appropriate OpenType tags.[9] The syntax for specifying features depends on the font-loader: with the built-in font-loader, you should use the standard "`+`⟨*tag*⟩" or "`-`⟨*tag*⟩" syntax, and when using fontspec as the font-loader, you can use that package's high-level interface.[10] 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 fontspec allows you to use commas.

Whenever you select a font for anything in this package, mathfont first checks whether your argument corresponds to a font family identifier in the NFSS, and if it does, mathfont uses that font.[11] Otherwise, the package loads the font, either through the built-in font-loader or with fontspec.[12] Advanced users should know that when the engine is LuaTEX, mathfont loads each font twice. During the first loading, mathfont makes no specifications for the renderer, so the first version of the font will typically use node mode. During the second loading, mathfont specifies base mode. My intention is that the font face with unspecified renderer is for text, and the font face with base mode is for math—using two different loading options provides the greatest access to OpenType font features throughout your document.[13]

You have three options for accessing blackboard-bold, calligraphic, or fraktur letters. First, the Unicode standard contains encoding slots for math alphanumeric symbols including these types of letters, and the last five keywords in Table 2 access this portion of the Unicode table.[14] If you call `\mathfont` on one of these ⟨*keyword*⟩s, the package defines the macro

---

[9] By 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.

[10] See the fontspec documentation for instructions on selecting OpenType features with fontspec.

[11] Specifically, if you use XETEX, mathfont uses the font name as given, and if you use LuaTEX, mathfont tries to use a font with family name ⟨NFSS *family*⟩`-base` instead. This is due to mathfont's double font loading and means that if you add fonts to the NFSS yourself in LuaTEX to use with mathfont, you should declare the font family twice. The first declaration can be normal, and for the second declaration, you should append `-base` to the family name. If you want to use OpenType font features in your equations, the second declaration should also instruct luaotfload to use `base` mode. If mathfont doesn't see ⟨NFSS *family*⟩`-base` in the NFSS in LuaTEX, it will print a warning and use the ⟨NFSS *family*⟩ instead.

[12] See `mathfont_code.pdf` for instructions on how to access the NFSS family name in this case.

[13] The luaotfload package supports two main modes for loading fonts: `node` mode is the default setting, and it supports full OpenType features in text but no OpenType features in math. The `base` mode supports fewer OpenType features, but the font features work in both text and math. Other loading options such as HarfBuzz are similarly useful for text but provide no access to OpenType features in math.

[14] The Math Alphanumeric Symbols block is U+1D400–U+1D7FF. A few blackboard-bold, calligraphic, and fraktur letters live in other areas of the Unicode table.

Table 4: Macros to Create Local Font-Change Commands

Command	Series	Shape
`\newmathrm`	Medium	Upright
`\newmathit`	Medium	Italic
`\newmathbf`	Bold	Upright
`\newmathbfit`	Bold	Italic
`\newmathsc`	Medium	Small Caps
`\newmathscit`	Medium	Italic Small Caps
`\newmathbfsc`	Bold	Small Caps
`\newmathbfscit`	Bold	Italic Small Caps

`\math`⟨*keyword*⟩`{`⟨*text*⟩`}`,

which behaves like a local font-change command from the next section and converts Latin letters into ⟨*keyword*⟩ style. For example,

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

defines `\mathbb` to typeset blackboard-bold letters using the glyphs from STIXGeneral. Second, you may have a Unicode font where the normal Latin letters are double struck, calligraphic, or fraktur, and in that case, you should declare a local font-change command using the tools in the next section. If you declare any of the macros `\mathbb`, `\mathcal`, `\mathfrak`, `\mathbcal`, or `\mathbfrak` this way, mathfont will ignore the corresponding keyword in future calls to `\mathfont`. Third, as of version 2.4, mathfont tries to be compatible with any macros `\mathbb`, `\mathcal`, `\mathfrak`, `\mathbcal`, or `\mathbfrak` that come from other packages or the kernel.

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 2. The eight commands in Table 4 accept a ⟨*control sequence*⟩ as their first mandatory argument and a ⟨*font name*⟩ as the second, and they define the ⟨*control sequence*⟩ to typeset any math alphabet characters in their argument into the ⟨*font name*⟩. You can specify OpenType features as part of the ⟨*font name*⟩ the same way as for `\mathfont`, described in the previous section. For example, the macro `\newmathrm` looks like

`\newmathrm{`⟨*control sequence*⟩`}{`⟨*font name*⟩`}`.

It defines the *control sequence* in its first argument to accept a string of characters that it then converts to the *font name* in the second argument with upright shape and medium weight. Writing

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

creates the macro

`\matharial{`⟨*argument*⟩`}`,

which can be used only in math mode and which converts the math alphabet characters in its ⟨*argument*⟩ into the Arial font with upright shape and medium weight. The other commands in Table 4 function in the same way except that they select different series or shape values. Finally, know that if you specify the font for Greek letters using `\mathfont`, macros created with the commands from this section will affect those characters, unlike in traditional LaTeX. Similarly, the local font-change commands will affect Cyrillic and Hebrew characters after you call `\mathfont` for those keywords.

Together these eight commands will provide tools for most local font changes, but they won't be able to address everything. Accordingly, mathfont provides the more general `\newmathfontcommand` macro. Its structure is

`\newmathfontcommand{`⟨*control sequence*⟩`}{`⟨*font name*⟩`}{`⟨*series*⟩`}{`⟨*shape*⟩`}`,

where the ⟨*control sequence*⟩ in the first argument again becomes the macro that changes characters to the ⟨*font name*⟩. You are welcome to use a system font name when you call `\newmathfontcommand`, but the intention behind this command is that you can use an NFSS family name for the ⟨*font name*⟩. 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. The commands from this section should appear in the preamble only.

4 Default Math Parameters

LuaTeX 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 the command

`\mathconstantsfont[`⟨*shape*⟩`]{`⟨*prev arg*⟩`}`,

where ⟨*shape*⟩ is an optional argument that can be "`upright`" (default) or "`italic`," and ⟨*prev arg*⟩ should be any argument that you have previously fed to `\mathfont`. When you call `\mathconstantsfont`, mathfont forces LuaTeX to always use the MathConstants table from the font that corresponded to that instance of `\mathfont` in the specified ⟨*shape*⟩. You don't need to set the MathConstants table when you use `\setfont` because the package calls `\mathconstantsfont` automatically in this case. This command will not work in XeTeX and should appear only in the preamble.

5 Lua Font Adjustments

The mathfont package provides six user-level commands to change bounding box, size, and accent positioning of characters in math mode. The command `\CharmLine` sets these features for a single math-mode character relative to its text-mode counterpart, and `\CharmFile` does the same thing for multiple characters at a time. (Charm stands for "character metric.") The argument of `\CharmLine` should be a list of integers and/or asterisks separated by commas and/or spaces, and Table 5 shows how many integers you need for different types of charac-

Table 5: Number of Integers Required in `\CharmLine`

Type of Character	Total Number of Entries
Latin Letters	5
Delimiters, Radical Sign (Surd Character), Big Operators	33
Everything Else	3

ters. The first integer should be the unicode encoding value of the character to be adjusted, and mathfont interprets the remaining numbers as follows.

- If the unicode value corresponds to a Latin letter, you should specify four more numbers. The next two integers tell LuaTEX how much to stretch the left and right sides of the glyph's bounding box when it appears in math mode, and the final two integers determine horizontal placement of top and bottom math accents respectively.
- If the unicode value corresponds to a delimiter, the radical (surd) symbol, or a big operator, you need to specify 16 pairs numbers, for a total of 32 more integers. The first 15 pairs are horizontal and vertical scale factors that mathfont uses to create large variants, where successive pairs determine the scaling of each next-larger glyph. The last two integers determine horizontal placement of top and bottom math accents respectively.
- If the unicode value corresponds to any other symbol, you should specify two more integers. They determine the horizontal placement of top and bottom math accents respectively.

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`. If you specify too few charm values, mathfont will raise an error, and if you provide too many, mathfont will silently ignore the extras. These commands should appear only in your document preamble.

For most applications, you can probably ignore charm information altogether, but 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. As is standard with decimal inputs in TEX, mathfont divides your inputs by 1000 before computing with them. Positive integers mean an increase, and negative integers mean a decrease. For a given character, the scale is usually the glyph width. For example,

```
\CharmLine{97, 200, -200, *, 50}
```

Table 6: Commands to Adjust Individual Characters

Command	Default Value	What It Does
`\RuleThicknessFactor`	1000	Thickness of fraction rule and radical overbar
`\IntegralItalicFactor`	400	Positioning of limits for integrals
`\SurdVerticalFactor`	1000	Vertical positioning of radical overbar
`\SurdHorizontalFactor`	1000	Horizontal positioning of radical overbar

Table 7: Lua Callbacks Created by mathfont

Callback Name	What It Does By Default
`"mathfont.inspect_font"`	Nothing
`"mathfont.pre_adjust"`	Nothing
`"mathfont.disable_nomath"`	Tell LuaTeX that we have a math font
`"mathfont.add_math_constants"`	Create a MathConstants table
`"mathfont.fix_character_metrics"`	Adjust bounding boxes, add character-specific math fields, create large variants
`"mathfont.post_adjust"`	Nothing

tells mathfont to take the lower-case "a" (unicode encoding value of 97), increase the bounding box on the left side by 20% of the glyph width, decrease the bounding box on the right side by 20% of the glyph width, do nothing to the top accent, and shift the bottom accent right by 5% of the glyph 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 particular font, consider making a custom set of charm values and uploading it to CTAN. Additionally, if you store your charm information in a file, you can read it in with `\CharmFile`. The argument of this command should be a file name, and mathfont reads the file and feeds each line individually to `\CharmLine`.

The commands in Table 6 adjust other aspects of the font as indicated. Each command accepts a single integer as an argument, and mathfont divides the input by 1000. With each macro, mathfont multiplies the quotient by some default length, so values greater than 1000 mean scale up, and values less than 1000 mean scale down. For example,

```
\RuleThicknessFactor{2000}
```

doubles the thickness of the fraction rule and radical overbar relative to the default, which varies between fonts. Changing the `\RuleThicknessFactor` is useful for fonts with particularly heavy or light weight. The `\IntegralItalicFactor` is important for making limits better fit integral signs, and the `\SurdVerticalFactor` and `\SurdHorizontalFactor` commands are essential when the top of the surd glyph differs from the top of its bounding box. The six control sequences from this section should appear in the preamble only.

Finally, advanced users who want to interact with the font adjustment process directly should use the six callbacks in Table 7. When luaotfload loads a font, mathfont (1) always calls `mathfont.inspect_font` and (2) calls the other five callbacks in the order that they appear in Table 7 if the font object contains `nomath=true`. 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 the `disable_nomath`, `add_math_constants`, and `fix_character_metrics` callbacks. 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.