Package mathfont v. 2.2 User Guide

Conrad Kosowsky

December 2022

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 objects.[1] The mathfont package loads TrueType and OpenType fonts for use in math mode, and this document explains the package's user-level commands. 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`. The mathfont installation also includes four example files, and all mathfont pdf documentation files are available on CTAN.

1 Loading and Basic Functionality

Loading fonts for math typesetting is more complicated than for regular text. First, the commands to select fonts for math mode, both in plain TeX and in the NFSS, are more complicated than the macros to select text fonts, and second, TeX expects fonts for math to contain extra information that it uses to format equations.[2] Broadly speaking, we say that a

Acknowledgements: Thanks to Lyric Bingham for her work checking my unicode hex values. Thanks to 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 their mathastext package.

[1] The last 30 years 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.

[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

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 equations that are less aesthetically pleasing 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] Please note that mathfont tries its best to get your fonts right, but if you run into trouble loading the correct font files, you should declare your font family and shapes in the NFSS before setting any fonts with mathfont. Because unicode text fonts greatly outnumber unicode math fonts, I hope that my package will expand the set of fonts available for typesetting math in LATEX.

You must use one of XELATEX or LuaLATEX to typeset a document with mathfont. You can load mathfont with the standard `\usepackage{mathfont}` syntax, and the package accepts three 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. 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 1.

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 the standard 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 this package instead of mathfont.[6] Users who want to a text font for math with pdfLATEX should consider Jean-François Burnol's mathastext because mathfont is incompatible with pdfTEX.[7] Finally, you will probably be better off using fontspec if your document does

boxes for letters in math mode—the Computer Modern *f* is a famous example. For this reason, mathfont also provides an interface to enlarge the bounding boxes of Latin letters when they appear in math mode. See section 4 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]In particular, 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://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.

Table 1: 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

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.

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 1, as in

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

and 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`

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

Table 2: 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

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 1 (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 1, it will ignore any future instructions to do so and prints a warning to the terminal instead.

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 2 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. For example adding "`onum=true`" tells TeX to load your font with oldstyle numbering, assuming that feature is present in the font.[9] Whenever you select a font, mathfont first checks whether you previously loaded fontspec, and if so, the package feeds your entire ⟨*font name*⟩ argument to fontspec. (You can also say "`fontspec`" as the ⟨*font name*⟩ to select the most recent font used by fontspec.) If you have not loaded fontspec, the package uses its own fontloader. I recommend letting mathfont handle font-loading because when using LuaTeX, mathfont takes care to load fonts in such a way that full OpenType features are accessible in text and limited OpenType features are accessible

[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.

in math. While it is also possible to do this in fontspec, it takes some doing.[10]

The last five keywords in Table 1 are a bit different. If you call `\mathfont` on a ⟨*keyword*⟩ from the last five rows in Table 1, the package defines the macro

`\math`⟨*keyword*⟩`{`⟨*text*⟩`}`

to typeset them. For example,

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

sets STIXGeneral as the font for bold calligraphic characters and defines `\mathbb` to access them. These are not for use with any double-struck, caligraphic, or fraktur font. Rather, they access Unicode's math alphanumeric symbols block. If you want to use a font where the regular letters appear double-struck, caligraphic, or fraktur, consider the font-changing control sequences in the next section.

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 1. The eight commands in Table 3 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*⟩. 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 3 function in the same way except that they select different series or shape values. Finally, know that if the user specifies 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 the user calls `\mathfont` for those keywords.

Together these eight commands will provide users with tools for most local font changes, but they won't be able to address everything. Accordingly, mathfont provides the more gen-

[10] 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 limited OpenType features, but the features will work for both text and math. When mathfont loads a font, it does so twice, once in `node` mode, which is primarily for setting the text font with `\setfont`, and once in `base` mode, which is for the package's other font declarations. This way you will be able to use OpenType features throughout your document.

Table 3: 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

eral `\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 with `\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 Table 3 as well as `\newmathfontcommand` 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 when you use `\setfont`. 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 positioning of characters in math mode. The commands `\CharmLine` and `\CharmFile` affect specific to various char-

Table 4: 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

acters. (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 4 shows how many integers you need for different types of characters. The first integer from the argument should be a unicode encoding number, and that tells mathfont how to handle the remaining values.

- If the unicode value corresponds to a Latin letter, 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. 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 will need to specify 16 pairs numbers, for a total of 32 entries. The first 15 pairs are horizontal and vertical scale factors that mathfont uses to create large variants, where successive pairs correspond to the 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, which will 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, but if you provide too many, mathfont will silently ignore the extras.

For most applications, you can probably ignore charm information altogether, but if you find bounding boxes or accent placement to be off slightly 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 typical with decimal inputs in TEX, mathfont divides your inputs by 1000 before computing with them. Positive integers mean "increase," and negative integers mean "decrease." For a given character, the scale is usually the glyph width. For example,

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

Table 5: 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 6: 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 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 5 adjust other aspects of the font as indicated. Each command accepts a single integer as an argument, and mathfont once again divides the input by 1000. With each of these macros, mathfont multiplies the quotient by some default length, so values greater than or less than 1000 mean "scale up" or "scale down" respectively. 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, and 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 6. When LuaTeX 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 6 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 behvaior associated with the callback, so do not mess with these three callbacks unless you are duplicating the default behavior of the callback or you really know what you're doing. Otherwise, you risk breaking the package. See `mathfont_code.pdf` for more information.