Package mathfont v. 1.6 User Guide

Conrad Kosowsky
December 2019
kosowsky.latex@gmail.com

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

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

Abstract

The mathfont package provides a flexible interface for changing the font of math-mode characters. The package allows the user to specify a default unicode font for each of six basic classes of Latin and Greek characters, and it provides additional support for unicode math and alphanumeric symbols, including punctuation. Crucially, mathfont is compatible with both XeLaTeX and LuaLaTeX, and it provides several font-loading commands that allow the user to change fonts locally or for individual characters within math mode.

Handling fonts in TeX and LaTeX is a notoriously difficult task. Donald Knuth originally designed TeX to support fonts created with Metafont, and while subsequent versions of TeX extended this functionality to postscript fonts, Plain TeX's font-loading capabilities remain limited. Many, if not most, LaTeX users are unfamiliar with the `fd` files that must be used in font declaration, and the minutiae of TeX's `\font` primitive can be esoteric and confusing. LaTeX 2_ε's New Font Selection System (NFSS) implemented a straightforward syntax for loading and managing fonts, but LaTeX macros overlaying a TeX core face the same versatility issues as Plain TeX itself. Fonts in math mode present a double challenge: after loading a font either in Plain TeX or through the NFSS, defining math symbols can be unintuitive for users who are unfamiliar with TeX's `\mathcode` primitive. More recent engines such as Jonathan Kew's XeTeX and Hans Hagen, et al.'s LuaTeX significantly extend the font-loading capabilities of TeX.[1] Both support TrueType and OpenType font formats and provide many additional primitives for managing fonts, and the fontspec package by Will Robertson and Khaled Hosny acts as a front-end for the font management built into these two engines.[2]

The mathfont package applies fontspec's advances in font selection to mathematics typesetting, and this document explains the package's user-level commands. Section 1 presents

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.

[1]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/`.

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

the basic functionality and related packages. Section 2 explains how to use the default font-change commands, and users in a hurry will find the most important information here. Section 3 describes the local font-change commands, and section 4 discusses mathematical symbols and aspects of their implementation. Section 5 addresses error messages. 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`. Both of these documentation files are included with the mathfont installation and are available on CTAN.

1 Basic Functionality

The mathfont package uses fontspec as a back end to load fonts for use in math mode, and it provides two ways to do this: (1) changing the default font for certain classes of math-mode characters; and (2) defining new commands that change the font locally for the so-called "math-alphabet" characters. The package can change the default math-mode font used for Latin, Greek, Cyrillic, and Hebrew letters; Arabic numerals; roughly 300 unicode math symbols; and standard unicode alphanumeric characters. The package accepts any OpenType or TrueType font, and tables 1 and 2 display the specific classes of characters that mathfont's default font-change command acts on. The default math-alphabet characters are Latin letters, Arabic numerals, upper-case Greek characters, and diacritics. When mathfont sets the default font for any of these four character classes, it preserves their math alphabet status, and when the package sets the default font for lower-case Greek, ancient Greek, Cyrillic, or Hebrew characters, it recodes each symbol in the class as math-alphabet type. At that point, the local font-change commands will act on any characters in those classes.

The package must be loaded with XƎLaTeXor LuaLaTeX. It can be loaded with the standard `\usepackage` syntax, and it accepts one optional argument. It treats the argument as a font name and changes all main fonts to that option. Specifically, the package invokes both `\mathfont` and fontspec's `\setmainfont`, and it defines the four local font-changing commands `\mathrm`, `\mathit`, `\mathbf`, and `\mathbfit` to produce text from the desired font in combinations of upright, italic, and bold styles according to the control sequences' last letters. XƎTeX users may run into trouble with fonts whose name contains multiple words because LaTeX eats spaces during package-option parsing. In this case, you will have to load the package and separately declare `\setfont` in your preamble. The package loads fontspec with the `no-math` option if and only if the user has not already loaded fontspec. Users who want fontspec without `no-math` or with other options in place can manually load it before requiring mathfont. Regardless, I strongly recommend that fontspec be loaded with `no-math` because otherwise some font changes may not render properly.

During loading, mathfont redefines three LaTeX internal macros to make symbol declaration compatible with unicode fonts, and default math-font changes work only with the redefinitions in place. Because the internal changes are relatively unobtrusive, mathfont's adjustments almost certainly do not affect LaTeX packages loaded later, and the package does not restore the commands automatically. Instead, `\restoremathinternals` returns the internal commands to their default definitions, and users who want the previous definitions should reset the kernel manually. The corollary is that `\mathfont` and `\setfont` work before `\restoremathinternals` but not after. As of version 1.6, the package optional arguments `packages`, `operators`, and `no-operators` are depreciated. Instead, mathfont offers `\restoremathinternals` as the only way to interact directly with the kernel.[3] For changes to big operators, use the `bigops` keyword in section 2.

The functionality of mathfont is most closely related to that of the mathspec package by Andrew Gilbert Moschou.[4] These two packages incorporate the use of individual unicode characters into math mode, and their symbol declaration process is similar. Both use fontspec as a back end, and both create font-changing commands for math-mode characters. However, the functionality differs in three crucial respects: (1) mathfont is compatible with LuaLaTeX; (2) it can adjust the font of basic mathematical symbols such as those in the first half of table 2; and (3) mathfont lacks mathspec's convenient space-adjustment character `"`.[5] Further, as far as I am aware, this package is the first to provide support for the unicode alphanumeric symbols listed in Table 2, even in the context of fonts without built-in math support. (Please let me know if this is incorrect!) In this way mathfont, like mathspec, is more versatile than the unicode-math package, although potentially less far-reaching.[6]

Users who want to stick with pdfLaTeX should consider Jean-François Burnol's mathastext as a useful alternative to mathfont.[7] This package allows the user to specify the math-mode font for a large subset of the ASCII characters and is the most closely related package to mathfont among those packages designed specifically for pdfLaTeX. Whereas mathfont works exclusively in the context of unicode fonts, mathastext was designed for the T1 and related encodings of Plain TeX and LaTeX. However, the mathastext functionality extends beyond

[3]To be clear, as of version 1.6, mathfont does not restore the LaTeX kernel when the user loads other packages. Given the scope and nature of the changes, I determined that the convenience factor of being able to use `\mathfont` anywhere in the preamble outweighs the incredibly small risk of interfering with another package. As far as I can tell, the biggest change is using a different primitive to code math symbols, but even that will probably never affect practical applications. The test `\if\mathchar\alpha` succeeds both before and after calling `\mathfont`, even though afterwards `\alpha` is defined with `\Umathchar` instead.

[4]Andrew Gilbert Moschou, "mathspec—Specify arbitrary fonts for mathematics in XeTeX," `https://ctan.org/pkg/mathspec`.

[5]Compatibility with LuaLaTeX comes at the expense of mathspec's space-adjustment character `"`, and spacing-conscientious users can either manually add `\kern` or `\muskip` to their equations or redefine an active version of `"`. For example, the code

```
\catcode`\"=\active
\def"#1{\ifmmode
  \kern⟨dimension⟩\relax #1\kern⟨other dimension⟩\relax
\else
  \char`\"#1%
\fi}
```

will serve as a hack that very roughly approximates mathspec's `"`. This code will redefine `"` to typeset a right double quotation mark in horizontal mode, but in math mode, the character will insert *dimension* and *other dimension* of white space on each side respectively of the next character. More advanced users can automate the dimensions by using TeX's `\if` or LaTeX's `\@ifnextchar` conditionals to test whether the following character needs a particular spacing adjustment.

[6]Will Robertson, "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 mistakenly stated that mathastext distorts TeX's internal mathematics spacing. In fact the opposite is true: mathastext preserves and in some cases extends rules for space between various math-mode characters.

that of mathfont in two notable aspects: (1) mathastext makes use of math versions, extra spacing, and italic corrections; and (2) mathastext allows users to change the font for the twenty-five non-alphanumeric characters supported by that package multiple times. After setting the default font for a class of characters, mathfont allows only the local font changes outlined in section 3.

2 Setting the Default Font

The `\mathfont` command sets the default font for certain classes of characters. Its structure is given by

`\mathfont[`⟨*optional character classes*⟩`]{`⟨*font name*⟩`}`,

where the *optional character classes* can be any set of keywords from Tables 1 and 2 separated by commas, and the *font name* can be any OpenType or TrueType font in a directory searchable by TeX.[8] The command loops through all keywords in the optional argument, and for each keyword, it changes the math-mode font for every character in that class to the *font name*.[9] Currently, mathfont does not support OpenType features in math mode. To change both math and text fonts simultaneously, the package provides the command

`\setfont{`⟨*font name*⟩`}`,

which calls both `\mathfont` and fontspec's `\setmainfont` using the *font name* as arguments. The package's optional argument is equivalent to calling `\setfont` and three local font-change commands from section 3, and most users will find this command sufficient for most applications. Both `\mathfont` and `\setfont` should appear only in the document preamble, i.e. before `\begin{document}`.

The user should specify any optional arguments for `\mathfont` as entries in a comma-separated list. The order is irrelevant, and spaces throughout the optional argument are permitted. The argument should contain no braces! Leaving out the optional argument will cause the command to revert to its default behavior, where it acts on keyword classes `upper`, `lower`, `diacritics`, `greekupper`, `greeklower`, `digits`, `operator`, and `symbols`. For example, if the user writes

`\mathfont{Arial}`,

mathfont will change the font of all Latin characters, Greek characters, diacritics, digits, operators such as log or sin, and `symbols` characters to Arial whenever they come up in math mode. The package provides control sequences to typeset many symbols that LaTeX does not include by default, and users gain access to these commands when they call `\mathfont` or `\setfont` with the appropriate keyword-option. In total, the package is capable of acting on some 800 unicode characters, and for a full list of symbols and control sequences, see `mathfont_symbol_list.pdf`, which is included in the mathfont installation and is available

[8]When specifying the *font name*, users need to input a name that fontspec will recognize and be able to load. Advanced users will note that `\mathfont` uses `\fontspec_set_family:Nnn` and therefore loads fonts in the same way as `\fontspec` and related macros from that package.

[9]These changes happen through LaTeX's `\DeclareMathSymbol`, and `\mathfont` is basically a very elaborately wrapped version of this command.

Table 1: Math Alphabet Characters

Keyword	Meaning	Default shape
`upper`	Capital Latin Letters	Italic
`lower`	Minuscule Latin Letters	Italic
`diacritics`	Diacritics	Upright
`greekupper`	Capital Greek Letters	Upright
`greeklower`	Minuscule Greek Letters	Italic
`agreekupper`	Capital Ancient Greek Letters	Upright
`agreeklower`	Minuscule Ancient Greek Letters	Italic
`cyrillicupper`	Capital Cyrillic Letters	Upright
`cyrilliclower`	Minuscule Cyrillic Letters	Italic
`hebrew`	Hebrew Letters	Upright
`digits`	Arabic Numerals	Upright
`operator`	Operator Font	Upright

on CTAN. Users can feed `\mathfont` a control sequence as its optional argument as long as the macro eventually expands to a comma-separated list of keywords and suboptions without braces.[10] Finally, `\mathfont` and `\setfont` will not change the default font for a class of symbols once one of them has already done so.

By default, mathfont will use one of an upright or italic shape for every character class, and users can override this setting by writing an `=` next to the keyword and either `roman` or `italic` following that. These two suboptions correspond respectively to an upright shape—normal shape in the language of the NFSS—and an italic shape. Table 1 includes the default shape-values for each keyword, and the package declares characters for all keywords in table 2 as upright by default. For example, the command

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

changes all math-mode Latin letters to Times New Roman with upright shape.

The package provides access to several types of letterlike symbols that appear frequently in mathematical writing, and the last five keywords in table 2 constitute these classes. Unlike with other keywords, mathfont doesn't create control sequences to access the symbols directly but rather defines a new command that converts letters into the appropriate style. When the user calls `\mathfont` with any of the last five keywords from table 2, the package both declares the appropriate unicode characters as math symbols and defines the macro

`\math`⟨*keyword*⟩`{`⟨*argument*⟩`}`

to typeset them. For example,

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

will set STIXGeneral as the font for bold calligraphic characters and define the command

[10] Technically, `\mathfont` expands its optional argument inside an `\edef`. When it scans an optional argument, mathfont temporarily converts spaces to catcode 9 and ignores them. However, if you feed `\mathfont` a macro with spaces in it, TeX has already scanned and tokenized those spaces, so we use `\zap@space` from the LaTeX kernel instead. Braces will wreck both this process and the `\@for` loop that comes later.

Table 2: Letter-Like and Other Symbols

Keyword	Meaning
`symbols`	Basic Symbols
`extsymbols`	Extended Symbols
`delimiters`	Parentheses, Brackets, and Braces
`arrows`	Arrows
`bigops`	"Big" Operators (see section 4)
`extbigops`	Extended "Big" Operators
`bb`	Blackboard Bold (double-struck)
`cal`	Caligraphic
`frak`	Fraktur
`bcal`	Bold Caligraphic
`bfrak`	Bold Fraktur

`\mathbcal` to access them in math mode. For the `bb` case, the associated command acts on Latin letters and Arabic numerals, and for the other four keywords, the associated command acts just on Latin letters. TeX will ignore and issue a warning in response to any other characters in the *argument*.

3 Local Font Changes

With mathfont, users can locally change the font in math mode by creating and then using a new control sequence for each new font desired.[11] The control sequences created this way function analogously to the standard math font macros such as `\mathrm`, `\mathit`, and `\mathnormal` from the LaTeX kernel, and the package provides four basic commands to produce them. Table 3 lists these commands. All four have the same argument structure: a control sequence as the first mandatory argument and a font name as the second. 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}
```

would create 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 three commands in table 3 function in the same way except that they select different series or shape values for the font in question. Table 3 lists this information. As of version 1.6, `\newmathbold` has been renamed to `\newmathbf` to put it in line with NFSS naming conventions.

[11]The five macros in this section are basically wrapped versions of LaTeX's `\DeclareMathAlphabet`.

Table 3: Font-changing Commands

Command	Font Characteristics
`\newmathrm`	Upright shape; medium weight
`\newmathit`	Italic shape; medium weight
`\newmathbf`	Upright shape; bold-expanded weight
`\newmathbfit`	Italic shape; bold-expanded weight

Together these four commands will provide users with the tools for almost all desired local font changes, but they inevitably will be insufficient for some particular case. Accordingly, `mathfont` provides the more general `\newmathfontcommand` macro that functions similarly to the commands from table 3 but allows for more general font characteristics.[12] Its structure is

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

where the control sequence in the first argument again becomes the macro that allows the user to access the specified font. The font name means any OpenType or TrueType font in a directory searchable by TeX, and the series and shape information refers to the NFSS codes for these attributes. Like `\mathfont` and `\setfont`, these commands should appear only in the document preamble.

Unlike the traditional `\mathrm` and company, `mathfont`'s local font change commands create macros that can act on Greek characters. If the user specifies the font for Greek letters using `\mathfont`, macros created with the commands from Table 3 will affect those characters; otherwise, they will not.[13] Similarly, the local font-change commands will act on Cyrillic and Hebrew characters after the user calls `\mathfont` for those keyword-classes.

4 Math Symbols

Choosing which unicode characters to recode is something of a delicate task because few unicode fonts contain more than the most basic math symbols. In designing this portion of `mathfont`, I attempted to find the largest set of characters that reliably appears in every or nearly every major unicode font, and I coded those characters in the `symbols` keyword. This keyword contains punctuation and common symbols such as $\pm$, $\div$, and ∞, and it will be sufficient for basic math typesetting. That being said, most math relies on a much broader collection of characters and arrows, and in other keywords, I coded every unicode math symbol that I could reasonably see being useful. The extended math symbols keyword `extsymbols` contains quantifiers, set and element relations, just about any binary relation you can imagine, and a few miscellaneous symbols such as `\sharp` and `\flat`. The `arrows`

[12]The package defines the four commands from table 3 in terms of `\newmathfontcommand`, and it specifies their style characteristics according to the kernel commands `\updefault`, `\itdefault`, `\mddefault`, and `\bfdefault`. Changing these macros will implicitly change the characteristics of the commands in table 3.

[13]LaTeX 2_ε defines lower-case Greek letters as `\mathord` characters, and `mathfont` changes this classification to `\mathalpha` type when it declares them as symbols. The local font change commands act only on characters of class `\mathalpha`, so these commands will act on lower-case Greek letters if `\mathfont` redefines them to be `\mathalpha`.

keyword contains a swath of hooked, curved, and bar arrows and even one lightning bolt arrow. Most standard unicode fonts don't contain many of those glyphs, and users who call `\mathfont` for a font without certain characters will see blank spaces in their final output instead of the corresponding symbols from `mathfont_symbol_list.pdf`. If this happens, check the `log` file because it will display any missing characters in your fonts.

It's worth emphasizing three aspects of mathfont's symbol declaration process. First, the package does not provide any symbols in and of itself but rather gives users access to symbols that already exist on their computers. This is why mathfont provides no additional symbols directly at loading and why some package commands can create blank spaces rather than their intended output. Second, mathfont's functionality currently does not include math symbols of variable sizes.[14] Recoded delimiters do not respond to `\left` and `\right`, and LaTeX replaces them with their original Latin Modern equivalents before rescaling appropriately. Thus `\mathfont` with the `delimiters` keyword will produce normally sized delimiters in the font of your choice and big delimiters in Latin Modern Roman. Similarly, big operators such as `\sum` and `\prod` appear normally sized instead of larger after setting their font with `\mathfont`. This is undesirable! I have isolated all delimiter and big operator characters in their own keywords, and I hope to address this limitation in future updates. Third, the package provides an extra comma character, similar to LaTeX's `\colon`.[15] TeX users have likely noticed the extra space surrounding commas in math mode, e.g. $10,000$ versus 10,000, and mathfont's `\comma` addresses this problem. Here the first ten thousand uses a standard `,` while the second uses `\comma`. As a rule of thumb, use `,` as a punctuation mark and `\comma` as a character separator.

5 Handling Errors

I have tried to make mathfont's error messages as clear as possible, and the help text will contain instructions for how to resolve the problem. Nevertheless, some of the possible error messages warrant additional explanation.

The most salient errors are the "Could not find fontspec" and "Missing XeTeX or LuaTeX" fatal errors. When the user loads mathfont, TeX must be able to find the package file `fontspec.sty`, and TeX must be operating under the XeTeX or LuaTeX engine. If either condition fails, TeX will stop reading in `mathfont.sty`.[16] As of version 1.6, mathfont's fatal

[14]Dynamic math-mode character sizing is a surprisingly thorny task. OpenType font designers specifically code certain characters to change size when they design the font, and LuaTeX's `\Udelimiter` and `\Umathoperatorsize` depend on this embedded feature. Because most unicode fonts come without resizing information, mathfont would have to manually add these settings to the LuaTeX font table. I intend to add this functionality in some future update, but I do not know what the timeframe looks like for those changes.

[15]Consider $\{x : x \neq 0\}$ versus $\{x\colon x \neq 0\}$. The first specification uses `:` while the second uses `\colon`. As a rule of thumb, use `:` for ratios and `\colon` as a punctuation mark.

[16]Note that mathfont doesn't actually determine the typesetting engine. Rather, it checks whether the XeTeX and LuaTeX primitives `\Umathcode`, `\Umathchardef`, and `\Umathaccent` are defined, so if for some reason these control sequences have definitions when the user loads mathfont with another engine, fontspec's more robust engine checks will take over and cause TeX to abort. The reasoning here is straightforward: mathfont verifies only that the current typesetting engine provides the commands that it directly needs, so its potential functionality remains as broad as possible. If fontspec becomes compatible with a third engine that also provides (analogues of) these primitives, there is no reason to prevent mathfont from working with

errors prevent TeX from reading in the rest of the `sty` file but do not crash the compilation process, and users who continue past one of mathfont's fatal error messages will see an "invalid command" error if they call a user-level command in their document. I designed these errors to be unobtrusive, and users can safely ignore them. Because of how mathfont performs its engine check, it is theoretically possible that users with very old XeTeX or LuaTeX distributions may see the second fatal error even when running one of these two engines, and the solution is probably to upgrade to a more recent version of the engine in question. Unfortunately, I do not know what the exact cutoff for XeTeX and LuaTeX versions is.[17]

The fontspec package includes a "`no-math`" option, and mathfont expects fontspec to be loaded with this option. As mentioned previously, mathfont loads fontspec by default, but users can load fontspec before mathfont if they want to manually specify the package options. Alternatively, LaTeX's `\PassOptionsToPackage` may be an even better way to proceed. If mathfont detects that fontspec was loaded without the `no-math` option, it will issue an error message saying so. This error is not paramount in the sense that the document will compile normally if a user ignores it, but mathfont will probably have trouble changing the font of certain math-mode characters in this situation. During development, Arabic numerals posed a particular challenge in this regard.

The "internal commands restored" error arises when the user calls `\mathfont` after the package already restored the small portion of the LaTeX kernel that it adjusts when loaded. Typically this happens when the user calls `\mathfont` after `\restoremathinternals`. The package will ignore any `\mathfont` commands in this situation, so while the error is technically harmless, you may not see some font changes you might have been expecting. Similarly, if the user tries to set the default font multiple times for some character class, the package will ignore any additional attempts, issue a warning, and continue the compilation process.

What should you do if you can't resolve an error? First, always, always make sure that you spelled all of your commands correctly and closed all braces and brackets. Then check the mathfont documentation—you may be trying to do something outside the scope of the package, or you may be dealing with a special case. The internet is a great resource, and websites such as the TeX StackExchange, ShareLaTeX, and Wikibooks' LaTeX wiki are often invaluable when dealing with TeX-related issues. Definitely ask another human as well! At that point you should email the author about your code—you might have identified a bug. I welcome emails about mathfont and will make every effort to write back to correspondence about the package, but I cannot guarantee a timely response.

that engine as well.

[17]However, the manual for a beta version of LuaTeX, v. 0.70.1, includes these primitives, so they are at least as old as May 2011. See
`https://osl.ugr.es/CTAN/obsolete/systems/luatex/base/manual/luatexref-t.pdf`