

Alternatives to Computer Modern Mathematics

Alan Hoenig

Department of Mathematics

John Jay College

445 West 59 St.

New York, NY 10019

(516) 385-0736 or (212) 237-8858

ajhjj@cunyvm.cuny.edu

Abstract

It is possible to generate hundreds of new math fonts using specially finagled math fonts produced by MetaFont to match Type1 PostScript fonts. This talk describes the *MathKit* project which enables authors ignorant of MetaFont, PostScript, and virtual fonts to create and use these fonts in a reasonably easy manner.

Introduction

I have long been impressed by the ingenuity and persistence of the \TeX community as its members have gallantly shown how \TeX can keep pace with all sorts of publishing needs and with all kinds of computer innovations, such as \TeX and the World Wide Web. But I have long been struck by one apparent gap in this effort — there is no good way to typeset mathematics if you want to use any of the beautiful Type 1 PostScript fonts instead of Computer Modern. It is common to see authors embed Computer Modern math in Times Roman, say, but CM math is really too spindly for such typesetting to be as good as we know \TeX is capable of. Several years ago, I wondered if there was a way to close this gap. One of the solutions I came upon is the subject of this talk. I'm particularly pleased by it because poky old METAFONT is an important component of this system. Perhaps *MathKit*, the name of my system, will help usher METAFONT into the next millenium.

MathKit is one attempt to deal with typesetting mathematics using fonts *other* than Computer Modern. Till now, authors have had few alternatives:

- They can use CM math together with a text font family such as Times Roman, but the result is not professional.
- They can use proprietary math fonts, such as MathTime or Lucida New Math, but that requires spending money.
- They can use the Euler math fonts, but these letterforms are a bit too idiosyncratic for some, and it is not well known how to properly implement them anyhow.

MathKit aids in the creation of math fonts which are compatible with a text font family — that is, it can help you typeset a Baskerville math document where the equations really look Baskerville-ish. Depending on your choice of parameters, you also get **bold** math fonts. *MathKit* consists of a perl script and some auxiliary files to help an author — even one ignorant of virtual fonts or of METAFONT — to perform these tasks.

What it does — a detailed look

MathKit takes METAFONT parameters that are appropriate to an outline font family and uses these to create new math fonts with METAFONT. The symbols and other special characters in these new fonts look pretty good — and are compatible with your outline fonts — but the italics and numerals look ghastly. Fortunately, that's not a problem. Using virtual fonts, we manufacture math fonts that combine the new special symbols (done by METAFONT that look pretty good) with letters and numerals from the outline fonts while we throw away all the ghastly stuff. *MathKit* does this work for you; it provides scripts for the remaining steps (all this is described below). It also provides style files for plain \TeX and for the NFSS of \LaTeX for you to use these fonts in your documents. *You don't need to know anything about METAFONT or virtual fonts to use MathKit and the resulting fonts.*

This version of *MathKit* comes with three sets of font templates. Since Times Roman and Palatino are so common, I have prepared templates for these fonts. For fun, I have also prepared a template for Monotype Baskerville. Times comes in regular and

Unbound Orbits: Deflection of Light by the Sun

Consider a particle or photon approaching the sun from very great distances. At infinity the metric is Minkowskian, that is, $A(\infty) = B(\infty) = 1$, and we expect motion on a straight line at constant velocity V

$$\begin{aligned} b &\simeq r \sin(\phi - \phi_\infty) \simeq r(\phi - \phi_\infty) \\ -V &\simeq \frac{d}{dt}(r \cos(\phi - \phi_\infty)) \simeq \frac{dr}{dt} \end{aligned}$$

where b is the “impact parameter” and ϕ_∞ is the incident direction. We see that they do satisfy the equations of motion at infinity, where $A = B = 1$, and that the constants of motion are

$$\mathcal{J} = bV^2 \tag{1}$$

$$E = 1 - V^2 \tag{2}$$

(Of course a photon has $V = 1$, and as we have already seen, this gives $E = 0$.) It is often more convenient to express \mathcal{J} in terms of the distance r_0 of closest approach to the sun, rather than the impact parameter b . At r_0 , $dr/d\phi$ vanishes, so our earlier equations give

$$\mathcal{J} = r_0 \left(\frac{1}{B(r_0)} - 1 + V^2 \right)^{1/2}$$

The orbit is then described by

$$\phi(r) = \phi_\infty + \int_r^\infty \left\{ \frac{A^{1/2}(r) dr}{r^2 \left(\frac{1}{r_0^2} \left[\frac{1}{B(r)-1+V^2} \right] \left[\frac{1}{B(r)-1+V^2} \right]^{-1} - \frac{1}{r^2} \right)^{1/2}} \right\}.$$

The total change in ϕ as r decreases from infinity to its minimum value r_0 and then increases again to infinity is just twice its change from ∞ to r_0 , that is, $2|\phi(r_0) - \phi'_\infty|$. If the trajectory were a straight line, this would equal just π ;

$$\Delta\phi = 2|\phi(r_0) - \phi_\infty| - \pi.$$

If this is positive, then the angle ϕ changes by more than 180° , that is, the trajectory is bent *toward* the sun; if $\Delta\phi$ is negative then the trajectory is bent away from the sun.

Figure 1: Here are Baskerville-like math fonts, produced by *MathKit*, together with Baskerville text fonts.

bold series, Palatino is regular only and Baskerville in regular and semibold.

However, I have had excellent luck matching one of the templates with a non-related text family. The Baskerville-like template works very nicely with Monotype Janson and Adobe Caslon, for example. Consequently, it is possible to generate not three new math font families, but *hundreds* of them, as the title to this document proclaims.

What you get as final output

MathKit itself produces lots and lots of scripts and batch files. Once these are properly executed you get the following:

1. Detailed instructions, both onscreen and in a small ASCII file, telling you how to proceed.
2. Virtual fonts for math and text typesetting. You will also get fonts for bold math if a template containing bold parameters is supplied.
3. Style files for plain T_EX and L^AT_EX (NFSS). These files support bold math if bold parameter templates were present.

What you will need

All files can be found on any CTAN or mirror site, unless otherwise noted.

1. First off, you will need current versions of T_EX and METAFONT. They must be sufficiently recent to support virtual fonts.
2. `fontinst`, version 1.5 or better. To install this software, retrieve *all* files from `fonts/utilities/fontinst/inputs` area.
3. For plain T_EX, Damian Cudgley's `pdcfse1` font selection macros are required. These can be found in `macros/plain/contrib/pdcmac`.
4. Perl needs installation as well: version 5 of Perl, a freely-available utility for all computer platforms and easily obtained from many computer archives and CD-ROM software collections. This is simply a matter of placing the perl executable somewhere on your computer's search path.

5. Your text fonts need to have been installed using Karl Berry's fontnaming conventions. Furthermore, these fonts must have been installed following the original T_EX encoding, often denoted as `OT1` or `ot1`.
6. Working copies of the T_EXware utilities `tftopl` and `vptovf`, which should already be part of your T_EX installation. Make sure both these executables are in some part of your computer's search path.

Installation

Installation of *MathKit* consists of three steps:

1. Create a directory called `mathkit`, and install all the *MathKit* files in it.
2. Create a work directory below `mathkit` and switch to this directory to do all your work.
3. Finally, there are a few parameters that need *careful* adjustment at the beginning of the file `mathkit.par`. Check the documentation for more details.

MathKit also makes it possible to typeset with some special font types, including blackboard bold, calligraphic, Fraktur, typewriter monospaced, and sans serif, and will provide typesetting commands for these fonts, provided the latter exist. Except for sans serif, though, you have relatively little choice in which kinds of fonts to install. These fonts and font sources are all available on CTAN. Here's what *MathKit* expects:

- *MathKit* uses the calligraphic alphabet in the Computer Modern symbol fonts.
- The typewriter font must be installed under the name `cmtt10` and you will need an outline form of this font.
- You will need the `eufm10` Euler Fraktur font in outline form for Fraktur typesetting.
- You will need the METAFONT source for Alan Jeffrey's blackboard bold fonts for blackboard bold typesetting. On CTAN, these can be found in the `fonts` area, or perhaps `fonts/bbold`.
- You have much greater freedom for sans serif fonts, as discussed above.

Executing the software

The main *MathKit* script requires three parameters at the command line:

Unbound Orbits: Deflection of Light by the Sun

Consider a particle or photon approaching the sun from very great distances. At infinity the metric is Minkowskian, that is, $A(\infty) = B(\infty) = 1$, and we expect motion on a straight line at constant velocity V

$$\begin{aligned} b &\simeq r \sin(\phi - \phi_\infty) \simeq r(\phi - \phi_\infty) \\ -V &\simeq \frac{d}{dt}(r \cos(\phi - \phi_\infty)) \simeq \frac{dr}{dt} \end{aligned}$$

where b is the “impact parameter” and ϕ_∞ is the incident direction. We see that they do satisfy the equations of motion at infinity, where $A = B = 1$, and that the constants of motion are

$$J = bV^2 \quad (1)$$

$$E = 1 - V^2 \quad (2)$$

(Of course a photon has $V = 1$, and as we have already seen, this gives $E = 0$.) It is often more convenient to express J in terms of the distance r_0 of closest approach to the sun, rather than the impact parameter b . At r_0 , $dr/d\phi$ vanishes, so our earlier equations give

$$J = r_0 \left(\frac{1}{B(r_0)} - 1 + V^2 \right)^{1/2}$$

The orbit is then described by

$$\phi(r) = \phi_\infty + \int_r^\infty \left\{ \frac{A^{1/2}(r) dr}{r^2 \left(\frac{1}{r_0^2} \left[\frac{1}{B(r)-1+V^2} \right] \left[\frac{1}{B(r)-1+V^2} \right]^{-1} - \frac{1}{r^2} \right)^{1/2}} \right\}.$$

The total change in ϕ as r decreases from infinity to its minimum value r_0 and then increases again to infinity is just twice its change from ∞ to r_0 , that is, $2|\phi(r_0) - \phi_\infty|$. If the trajectory were a straight line, this would equal just π ;

$$\Delta\phi = 2|\phi(r_0) - \phi_\infty| - \pi.$$

If this is positive, then the angle ϕ changes by more than 180° , that is, the trajectory is bent toward the sun; if $\Delta\phi$ is negative then the trajectory is bent away from the sun.

Figure 2: Palatino-like fonts with Palatino text.

1. The name of the parameter template: `tm` refers to Times-like parameters, `p1` to Palatino-like, and `bv` to Baskerville-like.
2. The name under which text fonts are installed. This is apt to be something like `ptm` or `mnt` for Adobe Times or Monotype Times New Roman, `pp1` for Palatino, and `mbv` for Monotype Baskerville (which is *quite* different from ITC New Baskerville). As mentioned above, though, you are welcome to any properly installed text font family as well. Simply specify its `fontname` abbreviation at the command line.
3. The encoding your fonts follow. Only `OT1` (original T_EX encoding) or maybe `ot1` are currently allowed. Use `ot1` if your system doesn't allow uppercase file names.

For example, I type

```
perl ../mathkit tm ptm OT1
```

in my work directory to create Times-like fonts following the original T_EX encoding. To create my Janson/Baskerville fonts, I type

```
../mathkit bv mjn OT1
```

at the command line.

Currently, you get bold math fonts *unless* you choose the Palatino-like `p1` template.

Making the fonts

The following steps complete the font creation. Perform them all within the *MathKit* work directory (Step 2 in the “Installation” section).

1. Execute the file `makegf.bat` to have META-FONT create your pixel fonts. This step will take some time.
2. You will need to compress all the pixel files. Inside UNIX, you can do this via a series of commands such as


```
foreach X (*.600gf)
foreach? gftopk $X $X:r.600pk
foreach? end
```

 Not all operating systems are so accommodating, so there is a file called `makepk.bat` which may be helpful in this regard. **Caution:** before executing this script, it will almost surely be necessary to edit it.
3. Execute the script `makepl.bat` to create some property list files needed by T_EX.

4. Run the file `makevp.tex` through T_EX. That is, execute the command `tex makevp` or something appropriate for your system. This step will take lots of time. Along with lots of superfluous files, this creates many virtual property list files with extension `.vpl`.

5. Create the actual virtual files by running every `.vpl` file through the program `vptovf`. You can do this easily in UNIX:

```
foreach X (*.vpl)
foreach? vptovf $X $X:r.vf $X:r.tfm
foreach? end
```

Even easier — execute the file `makevf.bat` that *MathKit* creates for you.

6. Test your fonts by processing `testmath.tex` (for L^AT_EX users) or `testmatp.tex` (plain T_EX) and then printing it. If adjustments are necessary, return to step 4 (‘run `tex makevp`) and proceed from that point onward. Adjustments to your fonts will be discussed below.

7. Only when you are completely satisfied with your new fonts should you execute the script `putfonts.bat`, which moves font files and style files to their proper places.

That still leaves behind files with extensions `.log`, `.mtx`, `.pl`, `.vpl`, `.bat`, `.600gf` (or something similar), and several other miscellaneous other files. You may safely delete all these.

Fine tuning and character adjustment The only adjustment that should be necessary are spacing adjustments to improve the appearance of over-the-character accents, subscripts, and character placement. The two test files that enable you test this are `testmath.tex` (for L^AT_EX) and `testmatp.tex` (plain). Run one of these files through T_EX and examine the printed output carefully. *MathKit* will have made two or more adjustment files for you that facilitate making changes to character spacing.

Font mongers note: you may be able to fine-tune the characters themselves by adjusting the parameter values in the *template* files to other than those provided. Feel free! If you find a particularly fine set of values different from what I have provided, I would be grateful if you passed them along to me.

Using your new fonts

MathKit produces two style files, one for L^AT_EX and one for plain. Their file names are formed according

Unbound Orbits: Deflection of Light by the Sun

Consider a particle or photon approaching the sun from very great distances. At infinity the metric is Minkowskian, that is, $A(\infty) = B(\infty) = 1$, and we expect motion on a straight line at constant velocity V

$$\begin{aligned} b &\simeq r \sin(\phi - \phi_\infty) \simeq r(\phi - \phi_\infty) \\ -V &\simeq \frac{d}{dt}(r \cos(\phi - \phi_\infty)) \simeq \frac{dr}{dt} \end{aligned}$$

where b is the ‘‘impact parameter’’ and ϕ_∞ is the incident direction. We see that they do satisfy the equations of motion at infinity, where $A = B = 1$, and that the constants of motion are

$$J = bV^2 \quad (1)$$

$$E = 1 - V^2 \quad (2)$$

(Of course a photon has $V = 1$, and as we have already seen, this gives $E = 0$.) It is often more convenient to express J in terms of the distance r_0 of closest approach to the sun, rather than the impact parameter b . At r_0 , $dr/d\phi$ vanishes, so our earlier equations give

$$J = r_0 \left(\frac{1}{B(r_0)} - 1 + V^2 \right)^{1/2}$$

The orbit is then described by

$$\phi(r) = \phi_\infty + \int_r^\infty \left\{ \frac{A^{1/2}(r) dr}{r^2 \left(\frac{1}{r_0^2} \left[\frac{1}{B(r)-1+V^2} \right] \left[\frac{1}{B(r)-1+V^2} \right]^{-1} - \frac{1}{r^2} \right)^{1/2}} \right\}.$$

The total change in ϕ as r decreases from infinity to its minimum value r_0 and then increases again to infinity is just twice its change from ∞ to r_0 , that is, $2|\phi(r_0) - \phi'_\infty|$. If the trajectory were a straight line, this would equal just π ;

$$\Delta\phi = 2|\phi(r_0) - \phi_\infty| - \pi.$$

If this is positive, then the angle ϕ changes by more than 180° , that is, the trajectory is bent *toward* the sun; if $\Delta\phi$ is negative then the trajectory is bent away from the sun.

Figure 3: Times Roman math + Times Roman text.

the naming scheme

```
z⟨mock-family⟩⟨font-family⟩
```

Here, $\langle mock-family \rangle$ is the two-character designation for one of the font parameter templates (such as `tm`, `pl`, or `bv`); the word ‘mock’ refers to the fact that these fonts imitate but don’t equal the actual fonts in this family. $\langle font-family \rangle$ is the Berry family designation. Thus, if I create a Times-like set of fonts for use with font family `ptm`, I would find files `ztmpm.sty` (\LaTeX) and `ztmpm.tex` (plain). In the same way, the style files for mock-Palatino and mock-Baskerville fonts are named `zplppl` and `zvbmbv` (with the appropriate extensions). Style files for my Baskerville/Janson math fonts have names beginning with `zvmjn`.

Plain \TeX At the top of your file, include the statement

```
\input ztmmtt
```

(or whatever the style file name is). Then, standard font nicknames such as `\it` and `\bf` and the math toggles `$` and `$$` will refer to these new fonts.

If bold fonts have been generated, a command `\boldface` typesets everything in boldface — prose, mathematics, whatever. Bold math may be appropriate for bold captions, sections heads, and the like. Like any other font-changing command, this command should be placed within grouping symbols.

\LaTeX and NFSS You simply need to include the style name as part of the list of packages that you use in the document. Thus, a typical document would have a statement like

```
\usepackage{ztmpm,epsf,pstricks,...}
```

at the outset.

If *MathKit* has created bold math fonts for you, a `boldface` environment will typeset everything in that environment as bold, including all mathematics.

Math support for other font families

The parameters for the font families are contained in files with names like `tm.mkr`, `tm.mks`, or `tm.mkb`. The extensions refer to “*MathKit* regular”, “*MathKit* semibold”, or “*MathKit* bold” sets of parameters. The current *MathKit* assumes that you will be creating at most one of the set of bold or semibold fonts but not both.

It was surprisingly easy to prepare these parameter files. I prepared a test document in which individual characters were printed on a baseline at a size of 750pt. It’s (relatively) easy to measure the dimensions of such large characters and METAFONT can be asked to divide by 75 to compute the

proper dimension for ten-point fonts. It was particularly easy for me to make these measurements as I used Tom Rokicki’s superior implementation of \TeX for NeXTStep. This package contains on-screen calipers, which take all the work out of this chore.

If you plan to create your own parameter files for other font families, please use the supplied files as models. Make sure all measurements are given in terms of sharped points `pt#`;¹ *MathKit* looks for this string. And please consider placing this information on CTAN.

Other details; in conclusion . . .

For additional information, please see my book, *TeX Unbound* Sample output using *MathKit*-tweaked fonts appears throughout this article. The current version of *MathKit* is in the [fonts/utilities/mathkit](#) area of CTAN. Additional details concerning *MathKit* can be found in the documentation file `mathkit.tex`, part of this package.

Interested authors may care to investigate the author’s companion package, *MathInst*, the current version of which appears in the [fonts/utilities/mathinst](#) area of CTAN. In case you have *MathTime*, *Lucida New Math*, *Euler*, or *Mathematica* math fonts, *MathInst* provides scripts for installing these fonts with text fonts of your choice.

This software is issued as is, subject to the usual GNU copyleft agreement.

If you have any questions, comments, or bug reports, please send them along to me.

References

- [1] Bouche, Thierry. “Diversity in Math Fonts.” *TUGboat* 19,2 (1998), pp.121–135.
- [2] Hoenig, Alan. *TeX Unbound: \LaTeX and \TeX Strategies for Fonts, Graphics, and More*. New York: Oxford University Press, 1999.
- [3] Hoenig, Alan. “Alternatives to Computer Modern Mathematics.” *TUGboat* 19,2 (1998), pp. 176–187.
- [4] Horn, Berthold. “Where Are the Math Fonts?” *TUGboat* 14,3 (1993), pp. 282–284.
- [5] Knuth, Donald E. *The METAFONTbook*. Reading, Mass.: Addison-Wesley, 1986.

¹ ‘Sharped points’ are “‘true’ units of measure, which remain the same whether we are making a font at high or low resolution” (*The METAFONTbook*, p. 33). See also pp. 32–35, 91–99, 102–103, 268, and 315 — all in *The METAFONTbook*.

Unbound Orbits: Deflection of Light by the Sun

Consider a particle or photon approaching the sun from very great distances. At infinity the metric is Minkowskian, that is, $A(\infty) = B(\infty) = 1$, and we expect motion on a straight line at constant velocity V

$$\begin{aligned} b &\simeq r \sin(\phi - \phi_\infty) \simeq r(\phi - \phi_\infty) \\ -V &\simeq \frac{d}{dt}(r \cos(\phi - \phi_\infty)) \simeq \frac{dr}{dt} \end{aligned}$$

where b is the “impact parameter” and ϕ_∞ is the incident direction. We see that they do satisfy the equations of motion at infinity, where $A = B = 1$, and that the constants of motion are

$$J = bV^2 \quad (1)$$

$$E = 1 - V^2 \quad (2)$$

(Of course a photon has $V = 1$, and as we have already seen, this gives $E = 0$.) It is often more convenient to express J in terms of the distance r_0 of closest approach to the sun, rather than the impact parameter b . At r_0 , $dr/d\phi$ vanishes, so our earlier equations give

$$J = r_0 \left(\frac{1}{B(r_0)} - 1 + V^2 \right)^{1/2}$$

The orbit is then described by

$$\phi(r) = \phi_\infty + \int_r^\infty \left\{ \frac{A^{1/2}(r) dr}{r^2 \left(\frac{1}{r_0^2} \left[\frac{1}{B(r)-1+V^2} \right] \left[\frac{1}{B(r)-1+V^2} \right]^{-1} - \frac{1}{r^2} \right)^{1/2}} \right\}.$$

The total change in ϕ as r decreases from infinity to its minimum value r_0 and then increases again to infinity is just twice its change from ∞ to r_0 , that is, $2|\phi(r_0) - \phi'_\infty|$. If the trajectory were a straight line, this would equal just π ;

$$\Delta\phi = 2|\phi(r_0) - \phi_\infty| - \pi.$$

If this is positive, then the angle ϕ changes by more than 180° , that is, the trajectory is bent *toward* the sun; if $\Delta\phi$ is negative then the trajectory is bent away from the sun.

Figure 4: MathKit makes possible math bold typesetting. Here is is Times Roman bold math + Times Roman bold text.