Alternatives to Computer Modern Mathematics

Alan Hoenig
17 Bay Avenue
Huntington, NY 11743
ajhjj@cunyvm.cuny.edu

TEX users early on hustled to do everything with TEX that was available to other publishing software. It has long been possible to use all kinds of graphics and fancy fonts with TEX. One hole remains in this scenario, and that is the ability to use fonts other than Computer Modern for scientific typesetting. This is a real puzzle, when you consider that the whole reason for TEX in the first place was scientific typesetting. In the accompanying discussion, we attempt to fill in this hole by presenting several strategies for non-CM mathematical typesetting.

Naïvely, you might wonder why we just can't replace the Computer Modern text fonts of a document by some other fonts for a brand-new look. If we combine Computer Modern math with Times Roman text (or something comparable), the variables look too anemic and thin compared with the text letters, and in an extended document this incompatible contrast between text and math grates on the reader. You may find that some differences between math and text are acceptable—after all, math *is* different from prose—but these variations are somehow too disparate.

This article, drawn from the more extended discussion found in chapter 10 of my book *TEX Unbound* (1998, Oxford University Press), contains the results of some experiments showing how to use TEX to generate technical documents using many handsome fonts. We will be creating series of virtual fonts to do the typesetting for us.

We will discuss several strategies for mathematical typesetting:

- replacing Computer Modern Roman by Monotype Modern;
- the use of commercial and other math fonts that can then be integrated with text fonts. Although vendors may supply macro and style files to perform this integration, we will explore virtual font approaches. The four special sets of raw math fonts include MathTime, Euler, Lucida New Math, and Mathematica fonts; and
- using variations of the usual Computer Modern bitmap math fonts whose parameters have been adjusted so they more closely match their accompanying text fonts.

Computer Modern math plus new text fonts

It may be typographically dangerous to willy-nilly change the text font of a document while retaining Computer Modern math, but it is possible to choose a font that does blend well with Computer Modern. Computer Modern fonts were designed using Monotype Modern No. 8A as a model. The digital font most resembling these fonts is Monotype's Modern font, widely available from digital font vendors.

You should install the fonts as per the usual procedures. Then, LATEX users should add the command

```
\renewcommand{\rmdefault}{mmo}
```

where `mmo` is the Berry name for the Modern font family. No changes need be made to the math font declarations, as we shall continue to use the Computer Modern math fonts.

New math raw fonts

In an ideal world, math fonts would be 100% compatible with text fonts. For the math fonts available to TEX, this statement is true only when Computer Modern fonts are selected, when Times Roman is used with MathTime or the Mathematica fonts, or when Lucida Bright is used with Lucida New Math. However, if reasonable compromises are permitted, a much wider selection of font matches is available.

The x-height is the dominant physical feature of a font, since so much of a document is lowercase. In our math fonts, we take pains to match the x-height of the math fonts with that of the text types. Moreover, it makes more sense to scale the math to the text, rather than vice versa, since there is almost always more text than math in a paper. Presumably, the 10-pt size for a text font is the optimal size for that font.

The MathInst utility

One way to install math fonts is via *MathInst*, written by me. *MathInst* creates an entire font environment for typesetting. At the moment, *MathInst* knows about four math fonts, the MathTime, Lucida, Euler math, and *Mathematica* math fonts. *MathInst* consists of a Perl script, and several additional files needed by *fontinst*. The main purpose of these Perl scripts is to match a designated text family with a set of math fonts to create new math virtual fonts.

MathInst produces lots of output. First are the fonts themselves which combine the given math fonts with a family of text fonts. In case an author has provided the names of other fonts, such as

- a sans serif *family* of fonts;
- a typewriter font;
- a calligraphic font;
- a fraktur font;
- an uppercase bold Greek font; and
- a blackboard bold font,

MathInst will make them available as well (the uppercase Greek bold font will be used to create bold math fonts). In all cases, *MathInst* takes great care to size all fonts against the text fonts to make sure that all lowercase alphabets are visually compatible.

It's not enough to be provided fonts—they must be integrated into a set of macros for easy use by an author. For LaTeX authors, *MathInst* provides a new package file that performs this integration and makes new commands available for the specialty fonts (Fraktur, blackboard bold, etc.) if these fonts are available.

Authors using `plain` will find a new style performing this same integration. These authors will need to make sure Damian Cugley's `pdcfsel` font selection macros are available. (They can be downloaded from the `macros/plain/contrib/pdcmac` area of any of the CTAN archives.)

Two test files are also provided—one for LaTeX and one for plain—so authors can see examples of the new font selection commands at work. Finally, a log file records information about the virtual font process, including scale factors, to make it easy to rerun the process using override values of the scale factors if necessary.

Installation *MathInst* itself is found on CTAN in the `fonts/utilities/mathinst` area. To install it, follow the detailed instructions that are part of the package. You will also need the `pdcfsel` font selection scheme for plain TeX mentioned above, and the text and math fonts themselves. You'll also need *fontinst*, version 1.5 or greater. Also, make sure the Perl executable appears on one of your system's path directories.

The *raw* math fonts consist of a series of outline font files plus the associated `afm` files. It's easy to install these fonts. Here are the steps appropriate for traditional systems. The same steps apply to TDS systems, but it will be necessary to be more specific about the paths for the files.

1. Place the math font files with the other scalable fonts.
2. Place the `afm` files with your other `afm` files.
3. Make sure a proper entry exists for each math font in the `map` file for your `dvi` postprocessor.

Only the last point requires additional comment. For example, for the *dvips* `psfonts.map` file for a traditional TeX system, we need entries like

```
%% MathTime fonts...
mtsy    MTSY    <mtsy.pfb
mtex    MTEX    <mtex.pfb
rmtmi   RMTMI   <rmtmi.pfb

%% Lucida New Math fonts...
lbma LucidaNewMath-Arrows <lbma.pfb
lbme LucidaNewMath-Extension <lbme.pfb
lbms LucidaNewMath-Symbol <lbms.pfb
lbmi LucidaNewMath-Italic <lbmi.pfb
lbmo LucidaNewMath-AltItalic <lbmo.pfb

%% Mathematica Math fonts...
Math1   Math1 <Math1.pfb
Math2   Math2 <Math2.pfb
Math3   Math3 <Math3.pfb
Math4   Math4 <Math4.pfb
Math5   Math5 <Math5.pfb
```

Note that these fonts are proprietary; please respect the licenses under which these fonts are sold or distributed.

Alone of the special math fonts, the Euler fonts are in the public domain. In the 1980s, the American Mathematical Society commissioned Hermann Zapf to draw a set of alphabets suitable for mathematical typesetting. The Society has since graciously made these beautiful alphabets available for free. The first major use of these fonts was to typeset the book *Concrete Mathematics*.

These fonts were implemented by METAFONT, and proper installation consists in placing the `tfm` and `pk` with their mates on your system. However, we will often be scaling these slightly to match various text fonts, and rather than regenerate many new bitmap fonts, it may be easier to use the scalable versions of these fonts, also available for free (courtesy of Basil Malyshev; they may be found in the `fonts/cm/ps-type1/bakoma` section of CTAN—but there are certain licensing conditions). In this case, these fonts will also need entries in your `map` file. These entries on a traditional TEX system should look something like

```
%% Euler fonts...
euex10   euex10  <euex10.pfb
eufb10   eufb10  <eufb10.pfb
eufb5    eufb5   <eufb5.pfb
eufb7    eufb7   <eufb7.pfb
eufm10   eufm10  <eufm10.pfb
eufm5    eufm5   <eufm5.pfb
eufm7    eufm7   <eufm7.pfb
eurb10   eurb10  <eurb10.pfb
eurb5    eurb5   <eurb5.pfb
eurb7    eurb7   <eurb7.pfb
eurm10   eurm10  <eurm10.pfb
eurm5    eurm5   <eurm5.pfb
eurm7    eurm7   <eurm7.pfb
eusb10   eusb10  <eusb10.pfb
eusb5    eusb5   <eusb5.pfb
eusb7    eusb7   <eusb7.pfb
eusm10   eusm10  <eusm10.pfb
eusm5    eusm5   <eusm5.pfb
eusm7    eusm7   <eusm7.pfb
```

Note that the Euler fonts come in a variety of weights and sizes; `m` and `b` represent medium and bold weights, and `f`, `r`, and `s` the fraktur, roman, and symbol fonts.

Installing text fonts The *MathInst* utilities expect the text font family (and the sans serif fonts too) to be installed using the Berry font-naming scheme. One way to do this is via my *vfinst* utility or via the PSNFSS files. These text fonts *must* be installed using the original OT1 TEX encoding.

Running MathInst Once the *MathInst* software has been properly installed, you execute a module by switching to a *MathInst* work directory like `mathinst/work` and issuing a command like

```
../mathinst mt mbv
```

This command installs a MathTime family of math fonts. If your computer doesn't seem to understand this command, issue the wordier incantation

```
perl ../mathinst mt mbv
```

instead. Then, follow the further instructions that appear on the computer screen.

New math virtual fonts with MathInst

MathInst uses the two-character designations

`mt`	MathTime
`lu`	Lucida New Math
`eu`	Euler
`ma`	Mathematica

to refer to the various math fonts. The new math font families use the *z naming convention*, whereby the font family name for the new math fonts uses the two-character math designations together with the text font family designation. Suppose we combine MathTime (`mt`) with Baskerville, whose family designation is `mbv` in the Berry scheme. The new fonts will be described in macro files `zmtmbv.tex` and `zmtmbv.sty` (for users of `plain` or LATEX). Baskerville plus Euler or Baskerville plus Lucida would form the z-names `zeumbv` and `zlumbv`. The z-name `zmambv` describes a marriage between Baskerville and the *Mathematica* math fonts.

The fonts themselves follow the Berry scheme, but you don't need to keep track of this, since the *MathInst* style files load the fonts for you automatically and establish their correspondence either with familiar nicknames (`\it`, `\bf`) or with the NFSS. But if you find yourself poking about your font directories, here's a quick key to the many new fonts you'll see.

Additional font variants `m`, `e`, `l`, or `a` indicate a math font, while variants `m`, `y`, or `v` following the encoding digit denote the math italic, symbol, or extension fonts. For Monotype Baskerville, the four math fonts at text size have names

- `mbvrm7t`, the math Roman font;
- `mbvrm7m`, the math italic font;
- `mbvrm7y`, the symbol font; and
- `mbvrm7v`, the extension font.

(The presence of the "7" in the font name reminds us we are using the original OT1 TEX encoding.) A math Roman font connecting Baskerville and Euler or Baskerville and Lucida would be called `mbvre7t` or `mbvrl7t`. A similar math Roman font for Baskerville plus Mathematica would be `mbvra7t`.

MathInst does its best to "fake out" new fonts at script and scriptscript sizes. You will see as many as three fonts for each of the above varieties listed above. In addition to `mbvrm7t`, the math Roman font for text sizes, you'll see `mbvrm7t7` and `mbvrm7t5`, the same fonts fine-tuned for use in script and scriptscript contexts. (The suffixes "7" and "5" recall the sizes of seven and five points that Computer Modern typesetting uses for these designations.) There is only a single math extension font for any family.

Using the new fonts Whether you use LATEX or plain TEX, you'll only need to remember the z-name for your new fonts. One purpose of the *MathInst* test files is to provide living examples showing just how the new fonts are invoked.

In a LaTeX document, all the work is done by the package file whose first name is precisely the *z*-name for the math fonts. As with all package files, its extension is `sty`. If you add a line like

```
\usepackage{zmtmbv}
```

following `\documentclass{...}` then all the usual LaTeX font-switching mechanisms will now apply to the `zmtmbv` fonts rather than to the default Computer Modern fonts.

To typeset mathematics in a `plain` document, simply place a statement like

```
\input zmtmbv
```

very near the beginning of your file. Thereafter, all the usual plain font commands like `\it`, `\rm`, `\tt`, `$`, `$$`, and so on, refer to the new math fonts. The `pdcfsel` package itself provides more flexibility than plain users are used to, and it would be well worth any (plain) reader's time to gain familiarity with this package.

The MathTime fonts Michael Spivak developed the MathTime math fonts to be used with the Times family of text fonts in a TeX document;Y&Y is the vendor. Many authors will find these fonts the most useful for math typesetting. Their "Times Roman-y" look goes well with many other Roman fonts.

The package consists of three fonts—an extension font, a math italic font, and a symbol font. The extension font `mtex` is directly analogous to `cmex10` (the same characters are in the same positions), but the remaining fonts have slightly different layouts from their Computer Modern counterparts. These differences are largely due to the elimination of the oldstyle digits and the calligraphic alphabet from the italic and symbol fonts. The slots opened up by these omissions have been filled with uppercase Greek letters and redesigned operator symbols. (The documentation that accompanies the fonts discusses the differences in greater detail.)

Users can also create MathTime math fonts in a second way—by following the instructions that accompany these fonts. This approach is not so heavily dependent on virtual fonts as is the *MathInst* way and relies on a well-written macro file accompanying this package.

The Euler fonts Euler fonts consist of math literals (neither Roman nor italic, but a unique upright font which is a compromise between the two forms), symbols (with a compatible uppercase calligraphic alphabet), Fraktur, and extension fonts. Because they predate virtual fonts, and because the font tables themselves follow slightly quirky layouts, they have not been as useful heretofore as they might have been. The extension font is quite sparse, but we can add virtual flesh using Computer Modern glyphs to fill in the blanks of the font table.

Lucida New Math The Lucida math fonts for TeX were designed by to follow normal TeX typesetting conventions and yet be compatible with the extensive Lucida and Lucida Bright font families, and are available for purchase from Y&Y.

This Lucida New Math family consists of five fonts. Because each contains the full complement of 256 characters, these fonts are crammed with all kinds of additional glyphs. These additions include all the special symbols that occur in the additional symbol fonts commissioned and made available by the American Mathematical Society and include a Blackboard Bold font. The three fonts useful for standard TeXnical typesetting are the symbol, math oblique, and math extension fonts. A math italic font doesn't follow the original math italic font quite as closely as the oblique font. Finally, a math arrow font contains many, many new symbols plus the Blackboard Bold alphabet.

Lucida math extension differs from other extension fonts in that it contains many new glyphs in its upper half. With this font properly in place, you can use a new extensible set of open brackets, additional wide accents, newly sized integral symbols, and a fully extensible integral. The font also contains the uppercase Greek alphabet, which we already use to construct the math Roman font. *MathInst* style files contain commands (where necessary) to use these new features.

The wide accent symbols are automatically in place; simply continue to use `\widetilde` and `\widehat` as before.

There are new kinds of square brackets that grow to enclose filler material. These brackets, are amenable to the usual "growth" mechanisms that govern `\left`, `\right`, `\bigl`, and so on.

The several new integral signs include new surface integrals, a new size for the regular and contour integrals, and pieces for a generally extensible integral. The new command `\surfint` and the existing integral commands `\int` and `\oint` work as expected. In addition, there are large variants, summoned into play by `\lint`, `\loint`, and `\lsurfint`. These control sequences ensure that the various integral signs change their size depending on a text or display context.

You might like to have TeX select the right size integral for you. For that reason, there are three variant integral commands, `\varint`, `\varoint`, and

`\varsint` (for regular, contour, and surface integrals) that try to do that for you. Each of these takes as argument the contents of the integrand. Figure 1 shows that this mechanism works poorly for the surface and contour integrals when the total height of the integrand is taller than the largest of the available integrals.

MathInst automatically places the TeX-hackery necessary in the Lucida style files it writes. You could type

```
$$
  \overbrace{\vphantom{\lint}
    \hbox{$\int\lint\oint\loint\surfint
    \lsurfint$\ }}%
    ^{\hbox{text}}
  \overbrace{\int\lint\oint\loint\surfint
    \lsurfint}%
    ^{\hbox{display}}
$$
$$\varint_{-\infty}^{+\infty}{\setlimits
  \left\Lbrack
\vcenter{\halign{\strut\hfil${#}$\hfil\cr
  \widehat 1\,\widehat{23}\,\widehat{456}\,
  \widehat{7890}\,\widehat{12345}
\cr
  \widetilde{67890}\,\widetilde{1234}\,
  \widetilde{567}\,
  \widetilde{89}\,\widetilde{0}
\cr
  \varoint{\short}\,\varoint{\med}\,
  \varoint{\tall}\,
  \varoint{\Tall}\,\varoint{\Talll}\,
  \varoint{\VTall}
\cr
  \varsint{\VTall}\,\varsint{\Talll}\,
  \varsint{\Tall}\,
  \varsint{\tall}\,\varsint{\med}\,
  \varsint_{\scriptscriptstyle\partial C}%
  {\setlimits\omega}
\cr
  \left\Lbrack x\right\Rbrack\
  \left\Lbrack\med\right\Rbrack
  \left\Lbrack \tall \right\Rbrack\
  \left\Lbrack \Tall \right\Rbrack\
  \left\Lbrack \Talll\right\Rbrack\
  \left\Lbrack \ontop{42pt}\right\Rbrack
\cr
  \varint_0^9{\Talll}\
    \varint_{-1}^{+1}{\Tall}\
  \varint{\tall}\
  \varint^{+\infty}_{-\infty}{\med}\
    \varint{x}\cr
}}\right\Rbrack\,dx
}$$
```

to get figure 1, which combines the Lucida math and Lucida Bright Roman fonts. Here, `\short`, `\med`, `\tall` and so on are simply temporary control sequences to generate arguments of various relative heights.

Mathematica math fonts The Mathematica math fonts were in development as this book was written, but Wolfram Research graciously made their interim fonts available to me. The fonts in their eventual release may have different names, different characters, and a different ordering.

These fonts consist of five font series comprising all of the characters that TeX normally expects, a calligraphic and a Blackboard Bold alphabet, and many more additional characters. Each series consists of four variants, normal, bold, typewriter, and typewriter bold. As far as TeX is concerned, the characters in these fonts are scrambled in a funny order, so we first create raw fonts, each of which appears more meaningfully ordered to TeX. You can do this with the script `makemma.tex`, part of *MathInst*. Running this script, and then creating virtual fonts in the usual way, creates three fonts `mmami`, `mmasy`, and `mmaex` (math italic, symbol, and extension), which are themselves suitable components for virtual font shenanigans. Although these three are in fact virtual fonts, we will treat them as raw fonts in the creation of additional virtual fonts.

Fine tuning the new math fonts

Adding special-purpose fonts Authors may want to add special fonts to their math style. Here's what *MathInst* allows you to add:

- a sans serif font *family*,
- a typewriter font,
- a blackboard bold font,
- a Fraktur font,
- a calligraphic font, and
- a bold Greek font (suitable for setting bold math).

You may add any, all, or none of these. If any of these fonts are present on your system, *MathInst* adds high-level font-switching commands to the style and macro files it creates that recognize the presence of these fonts.

Where do these fonts come from? Many of them are proprietary, but a large number of them reside in the public domain, albeit in unlikely or unsuspected places.

As far as typewriter type is concerned, I strongly recommend the freely available Computer Modern typewriter font `cmtt10`, which blends well with almost every other digital face. There are alternatives. The Pandora typewriter font `pntt10` is also free from CTAN, and of course the printer-resident Courier font is widely available. There are several other variants of `cmtt10` in the TeX suite that some users may prefer, and proprietary typewriter fonts

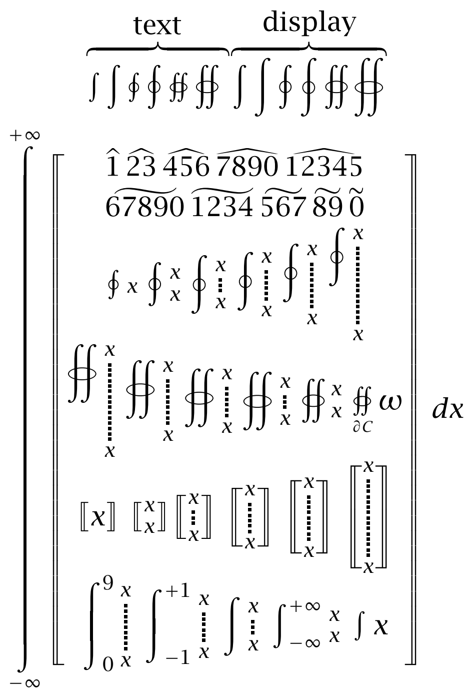

Figure 1: New Lucida math extension characters in action.

include offerings in the Lucida Bright families and ITC American Typewriter. Other authors may use a monowidth sans serif font (such as Letter Gothic) or some other contrasting face entirely.

A wide choice exists for sans serif families. Common choices will be Computer Modern sans serif, and the Helvetica fonts resident in all PostScript printers. I am personally partial to Gill Sans (from Monotype) and the Lucida Sans fonts (Bigelow & Holmes), but both of these are commercial fonts.

A calligraphic uppercase alphabet is necessary to make the `\mathcal` or `\cal` commands work properly. *MathInst* can add this alphabet (in a virtual way) to the math symbol font. Among the widely available candidates are alphabets from the Computer Modern symbol and Euler symbol fonts, and the printer-resident Zapf Chancery font. Several bitmap script fonts in the `fonts` area of CTAN (such as Calligra, the RSFS fonts, script fonts, and `twcal`) may be appropriate. Many commercial fonts are suitable, but authors should refrain from choosing too fancy a script.

There is less choice for a Blackboard Bold and Fraktur font. In CTAN, we find the `bbold` fonts (by Alan Jeffrey) and the `msbm` fonts developed and provided by the American Mathematical Society; both of these are in the `fonts` area of CTAN, the latter being in its `ams` subdirectory. Commercial sources include the Lucida New Math family (the "arrows" font contains the Blackboard Bold glyphs) and Adobe's Math Pi fonts (the sixth of these contains the Blackboard Bold). Choices on CTAN for fraktur include Euler fraktur `eufm` in `fonts/ams` (a scalable version is part of the BaKoMa collection, also on CTAN) and the `yfrak` fonts in `fonts/gothic/yfrak`. Commercial choices include the Math Pi package from Adobe (check out the second font in the series for Fraktur) and a font called Fraktur from Bitstream.

Mathematicians often want formulas in bold type. *MathInst* will create bold math fonts for you, but the sticking point might be bold variants of the uppercase Greek letters. Computer Modern, Euler, and Mathematica fonts contain bold Greek alphabets, but neither Lucida nor MathTime do. If a bold version of the Greek letters is available, *MathInst* would like to know about it. There seems to be nothing available that exactly matches the Lucida Greek types, but bold Greek Times fonts can be purchased.

To make these fonts visible to *MathInst*, you'll need to enter the names of the font files to the right of the equal signs in the statements making assignments to `$tt_`, `$sansserif_`, and so on. Don't forget to remove any comment characters from the beginning of the line! Thus, to use `pmp6` as the source for Blackboard Bold, we need the line

```
$bbold_ = "pmp6";
```

in the parameter `par` file. Note that font names in these statements need double quotes fore and aft. Note too that these changes need to be part of *all* the `par` files (or at least all the ones you'll be using).

MathInst produces test files `testmath.tex` for LaTeX and `testmatp.tex` for `plain`. These files show how to implement the fonts you've just created and exercises these fonts in some reasonably complete manner. The files themselves are closely modeled after a similar test file originally designed by Alan Jeffrey. (The original of this file appears on CTAN in the `fonts/utilities/fontinst` area.) It is a good idea to compile these tests and print one out each time you create a new math font family.

New math fonts via Metafont

Think of the reasons that Computer Modern math fonts clash with other text fonts—they are somehow too skinny, the wrong height and depth, and their shapes may not harmonize well with text fonts. Being that they are meta-fonts, can we not alter the parameters to generate math fonts that more closely approximate text fonts we may be using? This strategy lies behind the *MathKit* scripts I have developed. *MathKit* aids in the creation of math fonts that may be compatible with a text font family. It 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. This material can be found in the `fonts/utilities/mathkit` area of CTAN.

MathKit takes METAFONT parameters that are appropriate to an outline font family and uses these to create math fonts. The symbols and other special characters look pretty good—and are compatible with your outline fonts—but the italics and numerals look ghastly. Using TeX's virtual font mechanism, we create math fonts that combine the new special symbols with letters and numerals from the outline fonts. *MathKit* does some of this work for you, and provides scripts for the remaining steps (described in the accompanying documentation). It also provides style files for plain TeX and for the NFSS of LaTeX for you to use these fonts in your documents.

The current version of *MathKit* comes with three sets of font templates. Since Palatino and Times-Roman are so common, I prepared templates for these fonts. For fun, I prepared a template for Monotype Baskerville. Times comes in regular and bold series, and Baskerville in regular and semibold; Palatino is regular only.

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

1. Virtual fonts for math and text typesetting. You will also get fonts for bold math if you have supplied a template containing bold parameters.
2. Style files for plain TeX and LaTeX (NFSS) . These files support bold math if bold parameter templates were present.

The main *MathKit* script requires three parameters. These are:

1. The name of the parameter template. '`tm`' refers to Times-like parameters, '`pl`' 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, `ppl` for Palatino, and `mbv` for Monotype Baskerville (which is *quite* different from ITC New Baskerville).
3. The encoding your fonts follow. Only `OT1` or `ot1` (original TeX encoding) are allowed.

For example, I type

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

in my work directory to create Times-like fonts following the original TeX encoding. (If your system supports the `#!` syntax for specifying the name of an interpreter, then put the proper path at the very top of `mathkit`, make sure the execute bit is set, and type the simpler injunction `../mathkit tm ptm OT1` from the work directory.)

I've had success matching `bv` (Baskerville-like) parameters to other Roman fonts. For example, I typed

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

to generate a nice-looking set of fonts combining Monotype Janson text with Baskerville-like math fonts.

The following steps complete the font creation. Perform them all within the *MathKit* work directory.

1. Use the `mkdirs` script to create any missing directories.
2. Execute the file `makegf.bat` to have METAFONT create the pixel fonts for your fonts. This step will take some time.

3. You'll need to pack all the pixel files. The file called `makepk.bat` that may be helpful in this regard. *Caution:* before executing this script, it may be necessary to edit it.
4. Execute the script `makepl.bat` to create some property list files needed by the next step.
5. Run the file `makevp.tex` through TEX. That is, execute the command `tex makevp` or something appropriate for your system. This step will take some time. Along with lots of superfluous files, this creates many "virtual property list" files with extension `vpl`.
6. Create the actual virtual files by running every `vpl` file through the program `vptovf`; execute the file `makevf.bat` which *MathKit* creates for you.
7. Execute the file `putfonts.bat` to place the font and other files where they belong.

This sequence is summarized for you again on the computer screen when you execute *MathKit*.

Using the new fonts *MathKit* produces two style files, one for LATEX and one for `plain`. Their file names are formed according the naming scheme

`z`⟨*mock-family*⟩⟨*font-family*⟩

Here, ⟨*mock-family*⟩ 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. ⟨*font-family*⟩ 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 `ztmptm.sty` (LATEX) and `ztmptm.tex` (plain). In the same way, the style files for mock-Palatino and mock-Baskerville fonts are named `zplppl` and `zbvmbv` (with the appropriate extensions).

At the top of a `plain` file, include the statement

```
\input ztmmnt
```

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

If bold fonts have been generated, a command `\boldface` typesets everything in its way 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.

In LATEX documents, 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{ztmptm,epsf,pstricks,...}
```

at the outset.

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

If your outline fonts have been installed using expert fonts, you may need to alter the `\rmdefault` command. It might be necessary, say, to type

```
\rmdefault{ptmx}
```

instead of `\rmdefault{ptm}`.

Preparing parameter files It was surprisingly easy to prepare these parameter files. I prepared a test document in which individual characters are printed on a baseline at a size of 750 pt. 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 10-pt fonts. It was particularly easy for me to make these measurements, as I use 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 (those files with extensions `mkr`, `mks`, or `mkb`). Make sure all measurements are given in terms of "`pt#`"; *MathKit* looks for this string. And please consider placing this information in CTAN.

Rogues' gallery

The following displays show the results of mixing and matching various math families to many text fonts. *vfinst* installed all the text fonts, and *MathInst* or *MathKit* generated all the math + text fonts.

These displays should be regarded as experiments only. I showed these pages to several people, and all concluded that some of the experiments are successful and others are failures. However, no one agreed which were the successes and which were the failures!

Computer Modern math + Monotype Modern

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

$$b \simeq r \sin(\varphi - \varphi_\infty) \simeq r(\varphi - \varphi_\infty)$$
$$-V \simeq \tfrac{d}{dt}\left(r \cos(\varphi - \varphi_\infty)\right) \simeq \tfrac{dr}{dt}$$

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

$$J = bV^2 \qquad (1)$$
$$E = 1 - V^2. \qquad (2)$$

(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, $dr/d\varphi$ 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

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

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, $2|\varphi(r_0) - \varphi'_\infty|$. If the trajectory were a straight line, this would equal just π;

$$\Delta\varphi = 2|\varphi(r_0) - \varphi_\infty| - \pi.$$

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

Reprinted by permission of John Wiley & Sons, Inc. from Weinberg, *Gravitation and Cosmology* © 1972, John Wiley & Sons, Inc.

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

$$\begin{gathered} b \simeq r\sin(\varphi - \varphi_\infty) \simeq r(\varphi - \varphi_\infty) \\ -V \simeq \tfrac{d}{dt}\left(r\cos(\varphi - \varphi_\infty)\right) \simeq \tfrac{dr}{dt} \end{gathered}$$

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

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

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

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

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, $2|\varphi(r_0) - \varphi'_\infty|$. If the trajectory were a straight line, this would equal just π;

$$\Delta\varphi = 2|\varphi(r_0) - \varphi_\infty| - \pi.$$

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

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

$$\begin{aligned} b &\simeq r \sin(\varphi - \varphi_\infty) \simeq r(\varphi - \varphi_\infty) \\ -V &\simeq \tfrac{d}{dt}\left(r \cos(\varphi - \varphi_\infty)\right) \simeq \tfrac{dr}{dt} \end{aligned}$$

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

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

$$E = 1 - V^2. \qquad (2)$$

(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, $dr/d\varphi$ 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

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

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, $2|\varphi(r_0) - \varphi'_\infty|$. If the trajectory were a straight line, this would equal just π;

$$\Delta\varphi = 2|\varphi(r_0) - \varphi_\infty| - \pi.$$

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

Lucida New Math + Lucida Sans (Bigelow & Holmes; 8/10)

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

$$b \simeq r \sin(\varphi - \varphi_\infty) \simeq r(\varphi - \varphi_\infty)$$
$$-V \simeq \tfrac{d}{dt}\,(r\cos(\varphi - \varphi_\infty)) \simeq \tfrac{dr}{dt}$$

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

$$\begin{aligned} J &= bV^2 && (1)\\ E &= 1 - V^2. && (2) \end{aligned}$$

(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, $dr/d\varphi$ 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

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

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, $2|\varphi(r_0) - \varphi'_\infty|$. If the trajectory were a straight line, this would equal just π;

$$\Delta\varphi = 2|\varphi(r_0) - \varphi_\infty| - \pi.$$

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