Improving the Æsthetics of Mixed-Font Documents

Philip Taylor
Royal Holloway and Bedford New College, University of London, Egham Hill, Egham, Surrey, United Kingdom.
Tel: +44 784 443172 Internet: P.Taylor@Vax.Rhbnc.Ac.Uk

Abstract

The author has recently been involved in typesetting a 700 page book in a mixture of PostScript and Computer Modern fonts. This paper describes some of the techniques developed to improve the æsthetics of the book, with particular reference to the techniques necessary to integrate fonts from two distinct families. Reference is also made to techniques for improving the visual consistency of the book, such as direct editing of PostScript inserts and dynamic calculation of table widths.

During the period 1987 – 1989, I spent considerable time collaborating with Professor I.L. and Professor R.S. Gibson of the Universities of Waterloo and Guelph respectively on the typesetting of a major new work on the principles of nutritional assessment. The book was to be published by an internationally renowned publishing house, and the design was, to a certain extent, dictated by house standards, but the publishers allowed us considerable flexibility in the implementation of those standards, and the resulting work owes as much to original design as it does to house styles.

The trim and text dimensions of the book were fixed by the publisher as 36 pc × 55.5 pc and 27 pc × 45 pc respectively, and it was felt that these dimensions dictated a single-column style (there are also many unresolved difficulties in the implementation of multi-column styles, where floating and non-floating objects may span an arbitrary number of columns). We decided that the text font should be Adobe Times-Roman, with its variants Times-Italic, Times-Bold and Times-BoldItalic. The book was by no means maths-free, and rather than attempt to re-implement the maths-oriented definitions of `plain.tex` (or `lplain.tex`) we decided to retain the use of the Computer Modern Symbol, Maths Italic and Maths Extension fonts. Greek majuscules and minuscules were taken from the Adobe Symbol font, as upright rather than italic minuscules were desired.

Having chosen the fonts to be used, we next needed to decide on the most appropriate size and leading; this, to my mind at least, led to one of the most significant design decisions of the entire work. We experimented with ten, eleven and twelve point Times-Roman, on a variety of leadings, and felt that none of these offered the perfect combination of characters per line and lines per page. Ten-point Times-Roman had too many characters per line, eleven-point had just about the right number, and twelve-point had too few, while 10/12 point offered the ideal number of lines per page; 11/12 was too cramped vertically, 11/12.5 was about right but gave too few lines per page, and of course 11/13.6 and 12/14.5 gave even fewer. It seemed that we wanted the impossible: a font that was as wide as an eleven-point Times-Roman which could nonetheless be set on a twelve-point leading.

At this point, inspiration struck: as we were using PostScript fonts, on a PostScript output device (a Linotronic 300), there was no reason why we should not scale the fonts *anamorphically* — in other words, shrink the font vertically while leaving it unchanged horizontally. A few experiments (and a 'phone call to John Gourlay of ArborText, to find out how to retain the anamorphic scaling when their standard prelude insisted on turning it off again) later, we had a range of anamorphically scaled fonts available for proofing. It was immediately obvious that anamorphic scaling had to be subtle to be effective: 22/25 was awful; 23/25 OK, and 24/25 excellent. An anamorphic scaling of 24/25, applied to an 11/12.5 point font/leading, yielded a font that was a little over 10.5 points vertically, while of course still being eleven points horizontally, on a twelve-point leading. This gave us the ideal number of characters per line, lines per page, and adequate leading: 10.5/12 it was to be.

Of course, we still had to have the publisher's approval to use this font, so we carefully prepared sample pages of each of the potentially acceptable font/leading combinations, together with a sample

page in our new anamorphic 10.5/12 Times-Roman, and sent these off for approval. We were overjoyed when the publisher not only selected 10.5/12, but pronounced it "one of the best examples of Times Roman which he had seen."

Anamorphic scaling of POSTSCRIPT fonts was one thing, but anamorphic scaling of Computer Modern was another. We were not using Graham Toal's POSTSCRIPT outlines of Computer Modern, but 1270 dpi bitmaps (`pk` files); how were we to re-scale these? The solution was simplicity itself: rather than scale the fonts individually, we would scale the whole document. Fonts which are presented as POSTSCRIPT bitmaps are subject to exactly the same set of manipulations as any other POSTSCRIPT object, and thus can be anamorphically scaled. Of course, scaling of a Computer Modern font bitmap undoes at a stroke all the hard work that METAFONT and the font designer have together achieved, in terms of the exact pattern of pixels output by METAFONT for a particular character, design size, magnification and mode definition, but the amazing thing is, it just doesn't seem to matter! Admittedly the scaled Computer Modern fonts form only a small proportion of the total glyphs typeset, but it would take a very careful and trained eye to spot that they are not produced from the canonical bitmaps.

At this point, it is worth digressing slightly and discussing the implementation language(s). Of the two of us actually typesetting (rather then writing) the book, one (Ian Gibson) is a confirmed though flexible LaTeX user, while the other (the present author) is a totally bigotted Plain TeX addict, who will admit to no merit whatsoever in any of the so-called higher-level formats — needless to say, this lead to some interesting discussions... Nonetheless, it was amicably agreed that the book would be set in LaTeX, but that any changes to the default style files (and, as it turned out, to `lplain.tex`, `lfonts.tex` and `latex.tex`) would be implemented through the medium of Plain TeX. In practice, this led to very few problems. Just occasionally an obscure look-ahead involving `\futurelet` and/or `\afterassignment` failed unexpectedly, and this lead to a few problems in debugging the code, but in general Plain TeX and LaTeX co-existed satisfactorily without any real difficulties. POSTSCRIPT modifications were, of course, implemented in POSTSCRIPT.

The book also included a very large number of illustrations and graphs, neither of which are TeX's *forte*, and these were prepared using either Harvard Graphics or Adobe Illustrator. These packages are both capable of producing (encapsulated) POSTSCRIPT output, and of using the Adobe font set. In practice, the graphics packages were configured to use Adobe Helvetica, and captions typeset in TeX were similarly set in Helvetica. The mechanism by which these figures were incorporated is (very) loosely based on a macro which was first published as `\PrintChart` by ArborText in the documentation accompanying DVILASER/PS; the most recent version of the revised macro (`\PostScript`) is attached as Appendix A. The macro allows the coordinates of the desired portion of the original figure (with respect to the lower left corner of the page) to be specified, and also allows the final height and width to be specified — this does, of course, allow the aspect ratio of the figure to be altered during insertion, but this feature is rarely desirable or used. More important, however, is that it allows the inserted figure to be scaled up or down from the original, but this can have an adverse effect on the æsthetics and consistency of the book, for the following reason: the originals will almost certainly have been prepared without allowance for the possibility of subsequent scaling, and will typically have a consistent set of line sizes; after scaling, lines which were originally the same size (we chose 0.4 pt to match the default rules of TeX) would have been of varying size, and it was therefore necessary to edit the POSTSCRIPT files produced by Adobe Illustrator and Harvard Graphics to pre-scale the line thicknesses to allow for subsequent re-scaling. The font sizes similarly had to be pre-scaled by editing. This caused one amusing problem: the editor used by Ian to make these changes to the POSTSCRIPT files had an uncontrollable urge to write a `<control-Z>` at the end of each file, and a special POSTSCRIPT macro definition for `<control-Z>` had to be written, to prevent otherwise insuperable difficulties.

Returning to the question of fonts, the next issue to be addressed was visual font matching. Remember that we were using Adobe Times-Roman for the text, Computer Modern Maths Italic, Symbol and Extension for mathematics, and Adobe Symbol for Greek majuscules, minuscules and various other characters. The first problem was simply one of achieving consistent perceived size. While Adobe Times-Roman (scaled `\magstephalf`) and Computer Modern Maths Italic and Symbol tenpoint (scaled `\magstephalf`) are very well matched for size, we soon discovered that the Adobe Symbol font, similarly scaled, was far too large. The question was, by how much to reduce it? If design size was an inadequate criterion, perhaps the x-height of the font would be a better metric. We tried again,

this time adjusting the scaling of Adobe Symbol such that its x-height (as measured by `\fontdimen␣5`) exactly matched the x-height of Adobe Times-Roman; *still* the fonts appeared visually to be of different sizes. Finally we tried matching the heights of a pair of individual characters from the two fonts: as the μ from Symbol was the most common of the Greek minuscules, we matched its exact height to that of the character from the Times-Roman font with which it most frequently occurred in juxtaposition (lower-case g), and this technique was found to give the best visual match. The code for dynamic font matching is given in Appendix B.

A further problem manifested itself: in designing the Computer Modern fonts, Knuth had obviously been influenced by the requirements of mathematical typesetting. In particular, the height of maths operators such as plus, minus, greater-than, less-than, etc. had been optimised to match the heights of the lower-case letters between which they were most likely to occur. In a scientific (but not mathematical) text, maths operators tend to occur between numbers, upper-case letters and lower-case letters with approximately equal frequency, and when seen between numbers or upper-case letters, the maths operators tend to look rather too low. Barbara Beeton was consulted on this problem, and suggested that one obvious solution would be to use METAFONT to generate alternative glyphs for the maths operators, raising them by an appropriate amount from the baseline. Whilst this was undoubtedly the right approach, it was potentially very time-consuming, and a pure TEX solution was sought. We also wished to avoid making changes to the text of the document, and thus a solution involving the replacement of all occurrences of `+` by `\+`, `-` by `\-`, etc., together with re-definitions of `\+`, `\-`, etc., was ruled out. This left us with only one apparent option: the maths operators had to be made active and defined as macros, with their replacement text being that necessary to raise the original operator by an appropriate amount, while otherwise retaining the operator's original semantics including its maths class (`\mathord`, `\mathop`, `\mathbin`, `\mathrel`, `\mathopen`, `\mathclose`, `\mathpunct`). Rather than write an individual definition for each maths operator, we developed a single macro `\mathsraise`, which, when applied to a maths operator, made it active in maths mode and re-defined it in terms of its original meaning; we similarly developed an alternative form, `\raisemaths`, which had an analogous effect when applied to named maths operators such as `\pm`, `\ge`, `\le` and `\times`. We could then simply list the operators to be raised, as in:

```
\mathsraise <
\mathsraise >   %  also +, -, = and /
\raisemaths \pm
\raisemaths \ge %  also \le and \times
```

Thereafter each operator so listed had its new meaning in maths mode only, retaining its original meaning outside of maths mode. With hindsight, it might have been desirable to be able to specify individually the amount by which each operator was to be raised, but in practice a common factor of 0.15 ex seemed both necessary and sufficient. The `\mathsraise` macros are given in Appendix C.

Many other changes proved desirable in the production of this book; a new 'the-book' style was derived by extensive modification from the standard 'book' style for LATEX, and POSTSCRIPT-specific versions of `lplain.tex`, `lfonts.tex` and `latex.tex` were generated (called `pslplain`, `pslfonts` and `pslatex`, respectively). `PSlfonts`, in particular, was considerably modified from its progenitor, firstly to change all possible references to Computer Modern fonts to the nearest POSTSCRIPT equivalent, but subsequently to pre-load far fewer fonts, as the implementation placed severe restrictions on the number of fonts that could be concurrently loaded. Special `\struts` were defined for use in ruled tables, to ensure adequate vertical white space above and below each row, and automatic table sizing was implemented, which ensured that any table which exceeded 0.85 `\textwidth` was automatically stretched to occupy `\textwidth`, thereby improving the visual consistency of the book. This last technique required an iterative approach to circumvent the limitation imposed by `\multispan`, whereby all additional space is dumped in the last column spanned. A new version of `\caption` was implemented to make use of information saved by the `\PostScript` and `\SetTable` macros; this enabled us to set captions to the same width as the figure or table to which they referred.

All of this took time — considerable time — but in the opinion of the author it was time well spent. The finished book bears neither the traditional TEX stamp of Computer Modern, nor the traditional LATEX stamp of overly large fonts and excessive white space. It more closely resembles a traditionally typeset book, and this was, in essence, the underlying aim: the typography and design were intended to be totally transparent to the reader, serving solely to draw his or her attention to the content. We believe that we have achieved this aim.

Appendix A: The \PostScript macro, for the inclusion of PostScript figures

```
%
%       A very arcane TeX/PostScript macro
%       ==================================
%

\newif \ifoutline
\newdimen \border

\def \PostScript file:#1;
                 xmin:#2; ymin:#3; xmax:#4; ymax:#5;
                 width:#6; height:#7; options:#8%
{%
        \outlinefalse
        \border = 0 pc
        \setbox 0 = \hbox
               {\def    \file   {#1 }%
                \def    \xmin   {#2 }%
                \def    \ymin   {#3 }%
                \def    \xmax   {#4 }%
                \def    \ymax   {#5 }%
                \def    \width  {#6 }%
                \def    \height {#7 }%
                \def    \options{#8}%
                \def    \Xmean  {\dimen 0 }%
                \def    \Ymean  {\dimen 1 }%
                \def    \Xsize  {\dimen 2 }%
                \def    \Ysize  {\dimen 3 }%
                \def    \Width  {\dimen 4 }%
                \def    \Height {\dimen 5 }%
                \def    \0{\count 0 }%
                \def    \1{\count 1 }%
                \def    \2{\count 2 }%
                \def    \3{\count 3 }%
                \def    \4{\count 4 }%
                \def    \5{\count 5 }%
                \def    \6{\count 6 }%
%
% The options macros
%
                \def    \box {\outlinetrue}%
                \def    \outline {\outlinetrue}%
                \options
                \message {Border is \the \border}%
                \6 = \border
                \ifoutline
                        \message {Figure will be enclosed in a box}%
                        \def \rule
                               {\ifhmode
                                        \vrule
                                \else   \ifvmode
                                                \hrule
                                        \else
```

```
                                \message {Are you sure you
                                          should be trying to
                                          draw rules in this
                                          mode ?}%
                        \fi
                \fi
               }%
\else
        \message {Figure will not be enclosed in a box}%
        \def \rule
               {\ifhmode
                        \vrule width 0 sp
                \else   \ifvmode
                                \hrule width 0 sp
                        \else
                                \message {Are you sure you
                                          should be trying to
                                          draw rules in this
                                          mode ?}%
                        \fi
                \fi
               }%
\fi
\Xmean = \xmin
\advance \Xmean by \xmax
\divide \Xmean by 2
\0 = \Xmean
\Ymean = \ymin
\advance \Ymean by \ymax
\divide \Ymean by 2
\1 = \Ymean
\Xsize = \xmax
\advance \Xsize by -\xmin
\2 = \Xsize
\Ysize = \ymax
\advance \Ysize by -\ymin
\3 = \Ysize
\Width = \width
\4 = \Width
\Height = \height
\5 = \Height
\def    \xmean  {\the \0 }%
\def    \ymean  {\the \1 }%
\def    \xsize  {\the \2 }%
\def    \ysize  {\the \3 }%
\def    \width  {\the \4 }%
\def    \height {\the \5 }%
\def    \border {\the \6 }%
\vbox to \Height
       {\vss \rule
        \hbox to \Width
               {\hss
                \rule height 0.5 \Height depth 0.5 \Height
                \hskip 0.5 \Width
```

```
                                \special
                                       {ps::[asis,begin]
                                        0 SPB
                                        /ChartCheckPoint save def
                                        Xpos Ypos translate
                                        \width \xsize div
                                        \height \ysize div
                                         scale
                                        /sptobp
                                           {65536 div 72 mul 72.27 div} def
                                        \xmean sptobp neg
                                        \ymean sptobp neg
                                         translate
                                         newpath
                                        /border {\border sptobp} def
                                        \xmean sptobp \ymean sptobp moveto
                                        \xsize sptobp 2 div neg border add
                                        \ysize sptobp 2 div neg border add
                                         rmoveto
                                        \xsize sptobp border 2 mul sub 0 rlineto
                                         0 \ysize sptobp border 2 mul sub
                                                 rlineto
                                        \xsize sptobp border 2 mul sub neg 0
                                                 rlineto
                                         closepath
                                         clip
                                         newpath
                                         255 dict begin
                                        /showpage {} def
                                        }%
                                \special {ps:plotfile \file asis}%
                                \special
                                        {ps::[asis,end]
                                         end
                                         ChartCheckPoint restore
                                         0 SPE
                                        }%
                                \hskip 0.5 \Width
                                \rule height 0.5 \Height depth 0.5 \Height
                                \hss
                               }%
                         \rule \vss
                        }%
        }%
        \message {The dimensions of the figure are:
                        {ht: \the \ht 0}
                        {dp: \the \dp 0}
                        {wd: \the \wd 0}}%
        \copy 0
}
```

Appendix B: The \LoadFont macro, for the dynamic matching of font sizes

```
\newfam \symfam
\newcount \symfactor
\newcount \xheight
\newcount \fudgefactor
\newcount \scalefactor
\newcount \canonicalxheight

\def \loadfont #1=#2 [#3]
       {
        \fudgefactor = 10
        \scalefactor = 10
        \setbox 0 = \hbox {\the #3 0 g}
        \canonicalxheight = \ht 0
        \symfactor = 1000
        \font #1 = #2 scaled \symfactor
        \setbox 2 = \hbox {#1 \char '155}
        \xheight = \ht 2
        \ifnum \xheight = 0
        \then
                \relax
        \else
                \multiply \symfactor by \canonicalxheight
                \divide \symfactor by \xheight
                \multiply \symfactor by \fudgefactor
                \divide \symfactor by \scalefactor
        \fi
        \message {Loading font #1 = #2 scaled \the \symfactor}
        \font #1 = #2 scaled \symfactor
       }

\viiipt

        \loadfont \viiisymtext = pssym [\textfont]
        \loadfont \viiisymscript = pssym [\scriptfont]
        \loadfont \viiisymscriptscript = pssym [\scriptscriptfont]

\ixpt

        \loadfont \ixsymtext = pssym [\textfont]
        \loadfont \ixsymscript = pssym [\scriptfont]
        \loadfont \ixsymscriptscript = pssym [\scriptscriptfont]

\xipt

        \loadfont \xisymtext = pssym [\textfont]
        \loadfont \xisymscript = pssym [\scriptfont]
        \loadfont \xisymscriptscript = pssym [\scriptscriptfont]

\@addfontinfo \@viiipt {\textfont \symfam = \viiisymtext
                \scriptfont \symfam = \viiisymscript
                        \scriptscriptfont \symfam = \viiisymscriptscript}
```

```
\@addfontinfo \@ixpt {\textfont \symfam = \ixsymtext
                      \scriptfont \symfam = \ixsymscript
                              \scriptscriptfont \symfam = \ixsymscriptscript}

\@addfontinfo \@xipt {\textfont \symfam = \xisymtext
                      \scriptfont \symfam = \xisymscript
                              \scriptscriptfont \symfam = \xisymscriptscript}
```

Appendix C: The \mathsraise macros, for the height adjustment of maths operators

```
\catcode `\@ = 11               % make `@' a letter ...

\gdef \mathsraise               % define the main procedure with no parameters
{%                              % as we have to change catcodes immediately
    \begingroup                 % will be closed as first thing \m@thraise does
    \def \@ctivateall           % define a macro to make all characters
    {%                          % except <space> active
        \count 0 = 0                                    % for i := 0 to 127
        \loop   \catcode \count 0 = \active             % \catcode[i] := \active
        \ifnum  \count 0 < 127 \advance \count 0 by 1   % endfor
        \repeat \catcode 32 = 10\relax                  % \catcode <space> = 10
    }%                          % \@ctivateall now defined
    \def \m@thraise ##1%        % this procedure does all the real work, raising
    {%                          % the character while retaining its maths class
        \endgroup               % restore normal catcodes a.s.a.p.
        \mathchardef \t@mp = \mathcode `##1%            % get the mathcode of #1
        \mathcode `##1 = 32768\relax                    % make it maths-active
        \def \@ftermathchar ####1"{"}%                  % \mathchar" -> "
        \def \m@thselect   ####1% a procedure to select maths classes
        {%                      % the parameter is normally a hex string
                \count 0 = ####1\relax   % get the mathcode into \count 0
                \divide \count0 by 4096  % only interested in the top three bits
                \ifcase \count0 \mathord % use these to select the maths class
                        \or     \mathop
                        \or     \mathbin
                        \or     \mathrel
                        \or     \mathopen
                        \or     \mathclose
                        \or     \mathpunct
                        \or     \mathord
                \fi
        }%                      % \m@thselect now defined
        \edef \t@mp {\expandafter \@ftermathchar \meaning \t@mp}% gets hex value
        \edef ##1%              % define parameter-1 to yield the same character
        {%                      % raised by 0.15ex but retaining its maths class
                \noexpand \m@thselect {\t@mp}%
                {\mathchoice
                 {\raise 0.15ex \hbox {$\displaystyle \mathchar\t@mp$}}%
                 {\raise 0.15ex \hbox {$\textstyle \mathchar\t@mp$}}%
                 {\raise 0.15ex \hbox {$\scriptstyle \mathchar\t@mp$}}%
                 {\raise 0.15ex \hbox {$\scriptscriptstyle \mathchar\t@mp$}}}%
        }%                      % parameter-1 now defined
    }%
    \@ctivateall \m@thraise     % make everything bar <space> active, then
                                % get and process the next character
}%                              % \mathsraise now defined

\def \raisemaths #1%            % this procedure does all the real work, raising
{%                              % the character while retaining its maths class
   \def \aftermathchar ##1"{"}% % \mathchar" -> "
   \def \mathselect   ##1%      % a procedure to select maths classes
   {%                           % the parameter is normally a hex string
```

```
        \count 0 = ##1\relax     % get the mathcode into \count 0
        \divide \count0 by 4096  % only interested in the top three bits
        \ifcase \count0 \mathord % use these to select the maths class
        \or     \mathop
        \or     \mathbin
        \or     \mathrel
        \or     \mathopen
        \or     \mathclose
        \or     \mathpunct
        \or     \mathord
        \fi
    }%                           % \mathselect now defined
    \edef #1{\expandafter \aftermathchar \meaning #1}% gets hex value
    \edef #1%                    % define parameter-1 to yield the same character
    {%                           % raised by 0.15ex but retaining its maths class
            \noexpand \mathselect {#1}%
            {\mathchoice
             {\raise 0.15ex \hbox {$\displaystyle \mathchar#1$}}%
             {\raise 0.15ex \hbox {$\textstyle \mathchar#1$}}%
             {\raise 0.15ex \hbox {$\scriptstyle \mathchar#1$}}%
             {\raise 0.15ex \hbox {$\scriptscriptstyle \mathchar#1$}}}%
    }%                           % parameter-1 now defined
}%                               % \raisemaths now defined

\mathsraise <
\mathsraise >
\mathsraise +
\mathsraise -
\mathsraise =
\mathsraise /

\raisemaths \pm
\raisemaths \ge
\raisemaths \le
\raisemaths \times

\end % of text
```