SynthSlant – Synthetically Slanted Glyphs

Ch. L. Spiel*

v0.1a 2024/12/20

Abstract

Package synthslant provides macros to slant arbitrary glyphs in both directions. It can be used to fake a real slanted font for *short* pieces of text and it can generate startling effects, like, for example, upright italics.

* cspiel@users.sourceforge.org

Contents

Table of Contents continued on next page.

The font samples ›fga‹ on the title page were generated with the help of METAPOST using »URW Palladio« in styles ›roman‹ and ›italic‹. The affine transformations were `slanted .2` for the slanted roman and `slanted -.2` for the upright italics.

List of Tables

List of Figures

Quick Reference

Alphabetically sorted list of all user macros and environments defined by package synthslant. The list of all package options can be found on p. 6 and 7. The Index on pages 29 to 30 may provide some more detailed insights.

`\negslantcontext`
: Name of the microtype context used when typesetting backward slanted text. **11**

`negslantenvironment`
: Wrapper around `\synthslantbox` when slanting backward with `\textsynthuprightitalic`. **11**

`\slantcontext`
: Name of the microtype context used when typesetting forward slanted text (p. 11)

`slantenvironment`
: Wrapper around `\synthslantbox` when slanting forward with `\textsynthslant`. **11**

`\synthnegslant`
: Slant value used by `\textsynthuprightitalic`. **8**

`\synthslantbox{`⟨*slant*⟩`}{`⟨*text*⟩`}`
: Slant ⟨*text*⟩ (forward or backward) with ⟨*slant*⟩. **10**

`\synthslant`
: Slant value used by `\textsynthslant`. **8**

`\textsynthslant{`⟨*text*⟩`}`
: Forward slant ⟨*text*⟩. **9**

`\textsynthuprightitalic{`⟨*text*⟩`}`
: Backward slant ⟨*text*⟩. **10**

1 Introduction

The synthslant package provides a translator (e. g. LaTeX, pdfLaTeX, or LuaLaTeX) independent interface to shearing glyphs. It implements a generic operation where a short piece of text gets slanted forward or backward. Moreover, specialized macros for the two most important use cases are provided, namely slanting an upright font forward and making an italics font upright. Unbeknown to some users, pdfTeX performs a similar operation under the hood: of the 40,210 map lines in my *pdftex.map* currently 1,236 instruct pdfTeX to artificially slant a font. This means some three percent of the shapes are generated this way.

Similar transformations can be achieved by other means. I elaborate on one of the alternatives in Sec. 6 on p. 16. Package synthslant however focuses on ease of use and strict locality of the glyph manipulation.

1.1 Appeal for Artificially Slanted Type

Artificially slanted type have a lousy reputation. Whenever there is an order to round up the usual font suspects synthetically slanted, bolded[1], and condensed type along with artificial small-caps swiftly are stuffed into the black Maria.[2] I can retrace this condescension for synthetic bold and condensed variants. They spoil the glyphs' outline because they do not (and cannot) conserve the necessary proportions. For small-caps the problems are somewhat minor and I wonder how far one could get with an OpenType font that supports a `size` axis as well as an `opsz` axis in the necessary ranges to construct convincing small-caps out of the multiple-master font.

In my view artificial slanting keeps much of the font's character intact. In fact one accusation of synthetically slanted type is that is creates less contrast than a proper italic [14, p. 141] to which I object that less contrast can in fact be enough contrast in a particular setting. Moreover, small contrast with respect to the main type is a problem of second order. It does not devalue the shape *per se* as is true for artificial bold and condensed fonts.

What seems to have gotten lost in the discussion is the shapes of true italics that were designed alongside with the roman type. If we have an unbiased look at it – for example at the title page of this manual – the italic versions of the upright characters are so markedly different that I would like to ask whether they match the upright shape in a strict sense. For the double-storey ␣a␣ becomes single-storey, the start of the loop of ␣g␣ moves from the far left to the middle. Alongside, the aspect ratio of both of the counters change. These defy the common guidelines [13, Ch. 6] of font pairing. We can make sense of the seeming contradiction by recognizing that the italics shape is not simply slanted, but creates tension in respect to the upright type by a variety of additional design features. A famous quote of Zuzana Ličko applies once again:

1 Package amsbsy defines a »Poor Man's Bold« macro `\pmb` that works by ›overprinting‹. The authors of amsbsy recommend to prefer package bm for bold mathematical symbols, though.

2 See for example Ref. 14, p. 97, but compare p. 142 and also Ref. 9, p. 68n, for a more nuanced assessment.

> The most popular typefaces are the easiest to read; their popularity has made them disappear from conscious cognition. It becomes impossible to tell if they are easy to read because they are commonly used, or if they are commonly used because they are easy to read.

1.2 Some History

Italics accompanying a roman font date back to one of the earliest print shops, namely that of Aldus Manutius around 1500 A. C. Artificially slanted, also known as ›oblique‹, versions of upright fonts appear in the twentieth century, when type designers and foundries start to save time and money by automatically constructing a slanted version of a given roman type [9, p. 68n]. Synthslant closely follows on their steps.

Some fonts in current LaTeX distributions offer slanted series right out of the box. *Eureka!* In particular the oldest (and once upon a time the only) font family shipping with TeX, cm Roman – nowadays member of the cm-Super family – is available in a deluge of almost thirty shapes. It covers not just slanted roman or slanted smallcaps but also slanted typewriter and somewhat surprisingly upright italics[3]. Furthermore, the LaTeX 2_ε font selection scheme provisions ›`sl`‹ for slanted shapes and ›`ui`‹ for upright italics [6]. The former is accompanied by the macros `\slshape` and `\textsl`.

Refer to Tab. 1 on the right for a brief list of variable fonts[4] that offer a slant-axis[5] that can be controlled with fontspec's `Slant`[6] key and Tab. 2 for a rather incomplete list of fonts that are shipped with slanted shapes. For these fonts synthslant is largely superfluous unless e. g. they also come with an italics shape that is to be typeset upright.

Table 1: A short list of some variable fonts with a slant axis (`slnt`).

Font
Cairo
Commissioner
Geologica
Gluten
Inter
Recursive
Roboto Flex

It seems that the original idea of automatically shearing text in LaTeX to simulate a slanted shape goes back to David Carlisle who suggested to use the pdfTeX-primitive `\pdfliteral` for shearing [4]. Shortly thereafter Bruno Le Floch pointed to another pdfTeX-primitive, namely `\pdfsetmatrix`, available with (in 2013) more recent pdfTeX versions [16].[7] With the help of the latter affine transformations of arbi-

3 Only a few font families come with upright italics as, e. g., FF Seria, Joanna, Literata, Odile, or Romanée.

4 See also the LaTeX Font Catalogue for Fonts with OpenType or TrueType Support and search Google Fonts for families of variable fonts with a `slnt`-axis or fonts with an unusual variation at Variable Fonts.

5 The registered axis is called `slnt` and it is not to be confused with the `ital` axis.

6 Since fontspec version 2.9a as of 2024/2/13.

7 The user-level manipulation of the transformation matrix has been part of the pdf-standard since its initial publication in 1993 [2, Secs. 3.8 and 3.9] in the form of operator `cm` (›concat‹ – concatenate matrix to current transformation matrix). ¶ The primitive `\pdfliteral` was implemented already in the first release of pdfTeX in 1998 [15] and the primitive `\pdfsetmatrix` joined 2007 in pdfTeX version 1.40.0 [12].

TABLE 2: Selected fonts that come with their own slanted series. ¶ The table on the left-hand side shows serif fonts, the one on the right-hand side sans-serif fonts.

Font	Font
Arvo	Cabin
CM Roman	Clear Sans
Domitian	Cuprum
Droid Serif	Fira Sans
Erewhon	Gandhi Sans
Extended Charter	INRIA Sans
GFS Artemisia	Lato†
GFS Bodoni	Montserrat
GFS Didot	PT Sans
	Source Sans Pro

† The shape is activated with `\itshape`.

trary content can be coded directly by setting the transformation matrix. A slight variant of his code is used in this package for the PDF- and l3draw-slant engines. The implementations for PSTricks, Ti*k*Z, and fontspecfontspec are trivial as they build upon shear functions supplied by the respective packages.

1.3 Shear Transformation, Slant, and Angle

Mathematically the slant operation is a shear transformation, which can be expressed with the equation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & \sin\alpha \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \tag{1}$$

where the vector $(x, y)^{\mathrm{T}}$ is mapped to $(x', y')^{\mathrm{T}}$ and both are elements of the two-dimensional drawing plane $\mathbb{E}^2$. Compare with Figure 1.

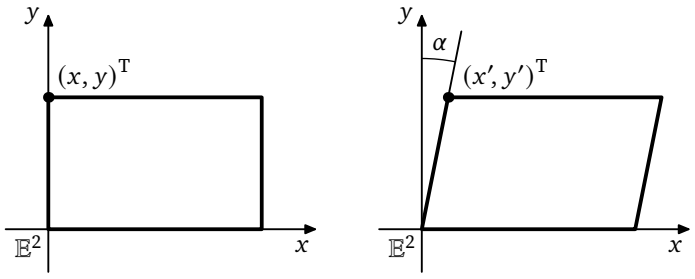

FIGURE 1: Shear transform of a rectangle by the angle α. The left-hand side shows the original figure the right-hand side the one sheared by α. The x-axis can be identified with the baseline of the text.

For $\alpha = 0$ the shear matrix becomes the identity matrix. Throughout of synthslant we work with the $\langle slant \rangle$ which is $\sin\alpha$ in Equ. 1 and avoid converting

back and forth to the shear angle α.[8] Some values for orientation: $\sin 5.74° \approx .1$, $\sin 11.5° \approx .2$, and $\sin 17.5° \approx .3$.[9] For real-life serif fonts $\langle slant \rangle$ is in the range of .1 to .45 and a value of .2 seems to be quite common. My *pdftex.map* lists negative $\langle slant \rangle$ values in the range $-.4$ to $-.05$ and positive $\langle slant \rangle$ values in the range .14 up to .45. See Table 3 on p. 8 for some actual values of serif fonts in LaTeX.

1.4 Usage Ideas

Automatic slanting both forward and reverse can be applied in a variety of typographic occasions. Here are some ideas.

1a. Generate a slanted serif in the unfortunate situation when a serif font comes without italics such as ›URW Antiqua‹.

Here, the user is relatively free to choose a $\langle slant \rangle$, for there are no italics whose angle must be matched. Synthslant's default of .2 should be a good starting point.

1b. If a secondary serif font – again assumed to have no italics or obliques – is paired with a primary serif font which has italics the slant angle of the former can be matched with that of the primary font. An example of such a constellation is ›Gentium‹ paired with ›Eczar‹.

3. Augment a serif font that features an italics shape with upright italics.

In nearly all cases it is desirable not to remove all forward-slant of the italics but retain some 1° to 2° of residual angle.

4. An italics shape that has an excessive slant angle, as e. g. ›Libre Caslon‹ may be corrected, i. e., partially un-slanted.

In this case, and generally if a font as a whole needs to be corrected, an alternative approach like the one sketched in Sec. 6 on p. 16 may be warranted.

5. Generate an oblique sans-serif if a sans-serif font comes without an oblique shape as, e. g., ›URW Grotesk‹.

6. Supply a slanted sans-serif shape for sans-serif fonts with designed, this is *true* obliques as e. g. ›Open Sans‹.

7. Fixed-width – also called ›typewriter‹ or ›teletype‹ – fonts without obliques (Yes, I am looking at you, Inconsolata!) finally get an oblique shape.

8. Small caps without accompanying italics can be slanted, too.

9. As synthslant also works in TeX's math-mode, it is possible to give math-italics even more of a heeling.

8 At least one slant engine currently requires such a conversion, namely PSTricks. The math is hidden from the user, though.

9 For small angles $|\alpha|$ measured in radians the sine is approximately linear: $\sin \alpha \approx \alpha$.

10. If the slant of the math script font is at odds with the slant of the usual math italics, it may be possible to apply synthslant on the script symbols for matching angles.

11. Big mathematical operators like the sigma can be slanted and others, like the integral sign, can have their inclination adjusted.

It is possible to obtain slants that run against the reading direction, so called ›backslanted‹ glyphs, but I have rarely seen an example where the typography of a document could benefit from that.

2 Package Options

```
\usepackage[⟨option⟩...]{synthslant}
```

This is a list of ⟨*option*⟩s that synthslant understands. The package options allow to predefine the forward and backward slant angles as well as the selection of a particular slanting engine.

If no ⟨*option*⟩s are given, synthslant assumes `auto` and `slant=.2`.

`auto`
: Let the package choose a slant engine. This is the default.

  For pdfLaTeX package synthslant selects the PDF-engine, for LuaLaTeX the fontspec-engine, and in all other cases the l3draw-layer handles the shear transformation.

`disable`
: Disable slanting completely.

`fontspec`
: Use fontspec as slanting back-end.[10]

`l3draw`
: Select the ›draw‹ layer of LaTeX3 as base for the slanting engine.[11]

  Caution

  This engine is experimental and the ›draw‹ layer of LaTeX3 itself is still experimental, too. See Sec. 5 on p. 15 for details. ■

`negslant=`⟨*slant-expr*⟩
: Set the default value for `\synthnegslant` *only*. The argument ⟨*slant-expr*⟩ is a floating-point expression. Note that for this option ⟨*slant-expr*⟩ must evaluate to a nonpositive value.

`PDF`, `pdf`
: Select the PDF-slant engine. This option requires that the document is translated with pdfLaTeX or a compatible program.

`posslant=`⟨*slant-expr*⟩
: Set the default value for `\synthslant` *only*. The argument ⟨*slant-expr*⟩ is a floating-point expression. Note that for this option ⟨*slant-expr*⟩ must evaluate to a nonnegative value.

`PS`, `ps`
: Use PSTricks to delegate slanting to the PostScript interpreter. Obviously requires PSTricks[12] and DVI-to-PostScript translation.

10 Requires *fontspec.sty*.
11 This option requires *l3draw.sty*.
12 The package actually required is *pst-3d.sty*.

Caution

This engine is still experimental and produces low-quality output! See Sec. 5 on p. 15 for details. ■

`slant=`⟨*slant-expr*⟩

Set the default values for both `\synthslant` and `\synthnegslant`, this is, act as if the two package options `posslant` = ⟨*slant-expr*⟩ and `negslant` = −(⟨*slant-expr*⟩) have been given. The argument ⟨*slant-expr*⟩ is a floating-point expression.

`TikZ`, `tikz`

Use TikZ for slanting.[13]

Caution

This engine is still experimental and produces low-quality output! See Sec. 5 on p. 15 for details. ■

The package options `slant`, `posslant`, and `negslant` all accept floating-point *expressions* as their arguments not just plain floating-point literals. See Ref. 7, Ch. 29, »The l3fp module – Floating points« for a description of the floating-point expression syntax and the available functions.

13 Requires *tikz.sty*.

3 Macros and Environments

This section describes how to actually apply the functionality of synthslant to some text. If the ⟨*slant*⟩ value matching a given font is known this is about it. To figure out an unknown ⟨*slant*⟩ value check out Sec. 4.

3.1 Variable-Like Macros

The amount of slanting forward (positive slant angles) and backward (negative slant angles) is controlled by two macros. They are set during package initialization. However, they can be changed at any time to accommodate for different fonts or special needs.

`\synthslant` Control the slant applied by `\textsynthslant`. This value is nonnegative.

```
\synthslant
```

To change the slant value to .24 say

```
\renewcommand*{\synthslant}{.24}
```

Table 3 summarizes some suggested slant values for selected fonts.

TABLE 3: Suggested slant values for selected *serif* fonts. The ⟨*slant*⟩ shown in the tables is not necessarily the one closest to the font's italics. Also compare with the left-hand table of Tab. 2.

Font	Slant	Font	Slant	Font	Slant
ADF Accanthis	.26	Crimson Pro	.2	Merriweather	.14
ADF Baskervald	.32	Crimson Text	.2	ML Modern	.23
ADF Berenis	.2	Day Roman	.2	Noto Serif	.22
ADF Venturis	.2	EB Garamond	.3	PT Serif	.2
Alegreya	.2	etbb	.2	Roboto Slab	.2
Arvo	.2	Faustina	.15	Quattrocento	.2
BaskervilleF	.2	fbb	.2	Source Serif Pro	.18
Bera Serif	.2	Garamond Expert	.2	Spectral	.18
Bitter	.16	Gandhi Serif	.2	STIX	.2
Brill	.22	Gentium	.2	TEX Gyre Pagella	.16
Caladea	.14	Ibarra Real Nova	.2	TX Fonts Serif	.2
Castoro	.18	IBM Plex Serif	.24	URW Antiqua	.2
Charis SIL	.17	INRIA Serif	.2	URW Nimbus Roman	.2
Clara	.24	Libertinus Serif	.2	Utopia	.2
Cochineal	.2	Libre Baskerville	.3	Vollkorn	.17
Coelacanth	.2	Libre Caslon	.38		

`\synthnegslant` Control the slant applied by `\textsynthuprightitalic`. This value is nonpositive.

```
\synthnegslant
```

3.2 Basic Interface

Package synthslant provides two easy-to-use macros for slanting glyphs. For a more flexible and powerful interface, see Sec. 3.3.

Note

The following restrictions and workarounds to get line-breaking and TEX's automatic hyphenation working again do *not* apply to the fontspec back-end. ■

Both macros provide *simplistic* support for slanting hyphenatable words and space-separated phrases for a given ⟨*text*⟩. The fundamental shear transformation would produce an single unbreakable horizontal box. So we have added two provisions to re-enable at least some breakability.

1. Spaces introduce breakpoint, e. g.

   ```
   \textsynthslant{topological dual space}
   ```

   slants the first word (producing a horizontal box) inserts a space and then slants the second word (producing another horizontal box). TEX sees two (unbreakable) boxes and a discardable space when it comes to linebreaking.

2. Discretionary hyphens in the form of ›\-‹ get propagated. So, we could improve on our above example by saying

   ```
   \textsynthslant{topo\-log\-i\-cal dual space}
   ```

   to ›recover‹ hyphenation of the first word.

This neither is a complete nor an elegant solution but it will take us quite far.

`\textsynthslant` Forward slant some upright glyphs.

```
\textsynthslant{⟨text⟩}
```

In horizontal mode switch to an upright shape, slant ⟨*text*⟩ with the slant value stored in `\synthslant` and apply »slant correction« – the equivalent of italics correction – at the right-hand side of ⟨*text*⟩.

In math mode just slant ⟨*text*⟩ with the slant value stored in `\synthslant`.

Tip

Discriminating typesetters will want to include trailing punctuation in ⟨*text*⟩. Compare for example:

```
\textsynthslant{FONT},
\textsynthslant{bar},
\textsynthslant{bay}.
```

FONT, *bar*, *bay*.

with

```
\textsynthslant{FONT,}
\textsynthslant{bar,}
\textsynthslant{bay.}
```

FONT, bar, bay.

This is similar advice as in the case of italics and also holds for `\textsynthuprightitalic` as well as any other slanting macro. ■

Use Cases

If italics seem to be too intrusive in the body we can substitute slanted text for example for foreign phrases like ›et. al.‹ and ›etc.‹:

```
\newcommand*{\foreignphrase}[2][USenglish]
                {\foreignlanguage{#1}{\textsynthslant{#2}}}
```

where we show the font modification in conjunction with the babel macro `\foreignlanguage` [3]. ¶

In math-mode you cannot have enough fonts, symbols, and most of the gizmos over there! I like to mark up automorphism groups associated with a given group with a slanted-roman typeface, though my macro has a more general name.

```
\newcommand*{\functionspace}[1]
                {\mbox{\textsynthslant{#1}}}
```
■

`\textsynthuprightitalic` Backward slant some italics or oblique glyphs.

```
\textsynthuprightitalic{⟨text⟩}
```

In horizontal mode switch to an italics shape, slant ⟨*text*⟩ with the slant value stored in `\synthnegslant`.

In math mode just un-slant ⟨*text*⟩ with the slant value stored in `\synthnegslant`.

Example

To set apart operators in an algebra like, e. g., the radical, we could use upright italics

```
\newcommand*{\algebraoperator}[1]
                {\mbox{\textsynthuprightitalic{#1}}}
```

and follow up with

```
\DeclareMathOperator{\rad}{\algebraoperator{rad}}
```

where we have assumed that amsmath has been loaded to bring `\DeclareMathOperator` into scope. ■

3.3 Advanced Interface

`\synthslantbox` Slant ⟨*text*⟩ with an amount of ⟨*slant*⟩ that can be positive, negative or zero.

```
\synthslantbox{⟨slant⟩}{⟨text⟩}
```

This is the unadorned call to the chosen slanting engine. In particular, neither the values of `\synthslant` nor of `\synthnegslant` enter its expansion! No corrections or TEX-mode adjustments are made.

Example

Generate a substitute for a missing solidus character:

```
\renewcommand*{\textfractionsolidus}[1]
  {\kern-.125em
   \raisebox{.125em}
            {\smaller
             \synthslantbox{.3}{\char'/}}%
   \kern.1em}
```

where the \smaller macro is from the relsize package [1]. ■

The following two environments are responsible for setting up everything before the actual slant or un-slant code runs and what happens after the slant-engine finishes. They can be redefined or patched to meet different users' needs.

slantenvironment (*env.*) Wrapper around \synthslantbox that is called for every forward-slanting operation with \textsynthslant.

```
\begin{slantenvironment}
  ...
\end{slantenvironment}
```

Switch to an upright font shape and – if package microtype [11] has been loaded – enter the Microtype-context defined by macro \slantcontext. At the end add some slant correction, which is the equivalent of italics correction.

***Use Cases* — »Patch Cases«**

Left-italics correction. ¶ Simultaneous left-italics and right-italics correction for a shift-left effect. ■

negslantenvironment (*env.*) Wrapper around \synthslantbox that is called for every backward-slanting operation with \textsynthuprightitalic.

```
\begin{negslantenvironment}
  ...
\end{negslantenvironment}
```

Switch to an italics font shape and – if package microtype [11] has been loaded – enter the Microtype-context defined by macro \negslantcontext.

\slantcontext Name of the microtype context used when typesetting slanted text.

```
\slantcontext
```

The expansion of this macro may be empty. The package's default is

```
tracking = synthslant
```

Note

The tracking context synthslant is *not* defined by synthslant. And microtype ignores undefined contexts. ■

\negslantcontext Name of the microtype context used when typesetting backward slanted text.

```
\negslantcontext
```

The expansion of this macro may be empty. The package's default is

```
tracking = synthnegslant
```

Note

The tracking context `synthnegslant` is *not* defined by synthslant. And microtype ignores undefined contexts. ■

Example

Upright italics often look somewhat tight. I like to add some extra tracking to them. So, I simply define the context `synthnegslant`:

```
\SetTracking[context = synthnegslant]
            {encoding = *, shape = it}
            {10}
```
■

Tip

When the tracking of upright italics is changed it may be advisable

- to break ligatures, e. g. `no ligatures = {f}`,
- to adjust the outer kerning, e. g. `outer kerning = {0, 0}` and
- to adapt the inter-word spacing, e. g. `spacing = {100,,}`.

The document *synthslant-gauge.tex*, which comes with package synthslant, has sample texts and tracking variations already set up for experimentation. ■

4 Determining Slant

If a synthetically slanted piece of text needs to match to an existing italics or oblique font the question arises how to determine the slant angle α or $\langle slant \rangle$.

Note

The slant angles of different glyphs in the same font may slightly differ from each other. Usually, longer shapes have less slant than short shapes.

We look for a representative $\langle slant \rangle$, a kind of average that achieves a visual match with the italics or obliques of the font family. ■

4.1 Direct Measurement

Measure the angle of some reference glyphs with a graphics program.

1. Prepare a page with some sample glyphs of the font shape to be matched.
2. Render it as PostScript or in PDF.
3. Load the file at a resolution of 1200 dpi or higher into your favourite graphics editor that supports measuring angles.
4. In the graphics editor center the interesting letters and set the zoom to one hundred percent or more.
5. Measure some letters and write down the angles.
6. Convert the desired angle α to a $\langle slant \rangle$ by calculating $\sin \alpha$.
   If no computer is available, the following formula might help:

$$\langle slant \rangle = \sin \alpha \approx \frac{11}{630^\circ}\, \alpha,$$

   where α is given in degrees.

4.2 Comparison of Shapes

Compare some reference glyphs with a differently slanted versions.

1. In file *synthslant-gauge.tex* which comes with the synthslant package insert the code to load your font-of-interest.
2. Render the document as PostScript or as PDF.
3. Load the first page at a resolution of 600 dpi to 900 dpi into your favourite graphics editor.
4. Cut the italics sample at the top allowing for generous white-space around it as a rectangle
5. Paste the rectangle in a new layer called e. g. ›sample‹.
6. On layer ›sample‹ move the rectangle down the list of different slant values until it match best.
7. Switch the layer mode of ›sample‹ to ›difference‹ and fine-position the rectangle over the slanted sample. Compare different letters in that way. Change line until the best match is found.

8. Read the slant value at the left-hand side of the line. See Figure 2.

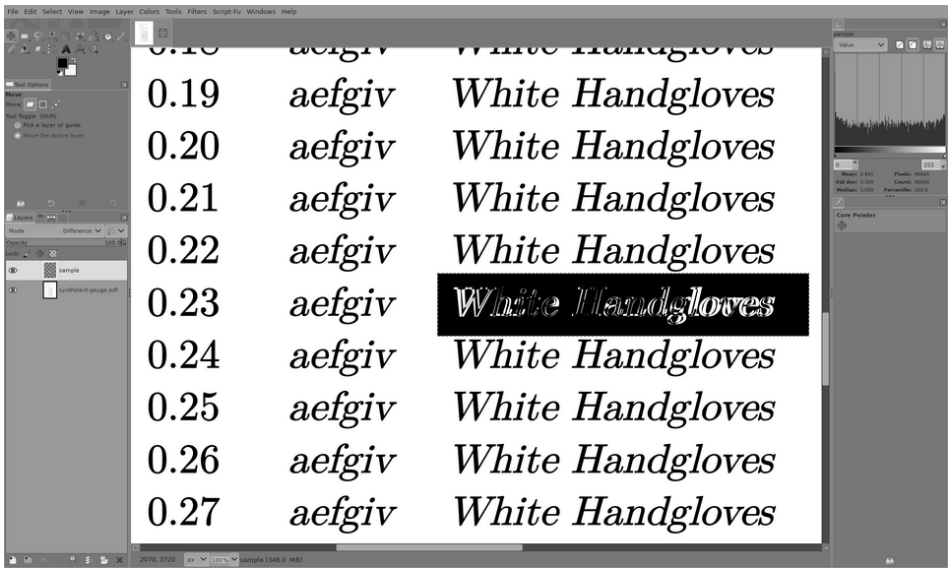

FIGURE 2: Compare italics and slanted samples with The Gimp. ¶ For this screendump I loaded the samples on the first page of *synthslant-gauge.pdf* at a resolution of 600 dpi into The Gimp. The ›sample‹ layer is aligned to the letter ›t‹ in the word ›White‹. Note that accidentally the letter ›H‹ of the next word ›Handgloves‹ confirms the good match.

4.3 Exploring Further

Once a usable slant value has been found it can be fed into *synthslant-gauge.tex* and – after recompiling with the appropriate LaTeX-engine – used to examine the details of the slant operations.

Page 2, Sec. 3.1, ›Copy‹, shows wild mixes of different font shapes, native and synthesized ones. Here, the slanted glyphs as well as the upright italics should blend well with the native italics/obliques and with the normal font, respectively.

Page 3, Sec. 3.2 and following sub-sections, examine the coupling of synthslant with the TeX-system and some of its extensions. If a slant engine malfunctions, it will become evident on this page.

5 Limitations and Known Problems

Here I list some of the known problems of syntslant. Conceivably there are more.

All except fontspec. Syntslant manipulations may not survive (pre-)processing by METAPOST.

l3draw engine.

- Depending of the shear direction the l3draw engine may generate some extra positive or negative space at the ends of the text.
- Any box sheared looses its depth; technically `\dp` becomes 0pt.
- Markedly slower than the PDF-implementation!

PSTricks engine. The PSTricks engine produces some extra space at the ends of the text.

Ti*k*Z engine. The Ti*k*Z engine produces some extra space at the ends of the text.

6 Alternative Solution

Here is an alternative to synthslant that I am aware of. It changes the slant of a font as a whole and it is impossible to undo the change within the document.

6.1 Using pdfTEX

In pdfTEX fonts can be re-mapped in the document preamble with the primitive `\pdfmapline`; see the pdfTEX Reference Manual [16, Sec. 6.1] for a description of the syntax. This possibility renders possible to splice in a slanting operation on the fly.

Here is a simplified syntax of a font map line, which does not indicate any of the optional parts for better readability:

```
⟨tfm-name⟩ ⟨ps-name⟩ ⟨font-flags⟩
           "⟨special⟩" <⟨encoding-file⟩ <⟨font-file⟩
```

where

- ⟨*tfm-name*⟩ is the basename of the TEX font-metric file (**.tfm*),
- ⟨*ps-name*⟩ is the name the font will acquire inside of TEX,
- ⟨*font-flags*⟩ optionally specify some characteristics of the font,
- ⟨*special*⟩ prescribes font manipulations in the same way as **dvips** [10, Sec. 6.3] does,
- ⟨*encoding-file*⟩ is the filename (**.enc*) where the encoding to be used with ⟨*font-file*⟩ is stored, and
- ⟨*font-file*⟩ sets the filename of the font's definition. It is given without path but includes an extension, which typically is *otf, pfb,* or *ttf.*

We are particularly interested in the ⟨*special*⟩ part that allows us to slant the whole font with a single instruction.

I want to elaborate the example given in Sec. 1.4, item 4 and generate less-angled italics for Libre Caslon. Here is a suitable map line taken from *pdftex.map* on my system:

```
LibreCsln-Italic-osf-t1-base LibreCsln-Italic
    "␣AutoEnc...␣ReEncodeFont␣"
                <[lcsln....enc <LibreCsln-Italic.pfb
```

which I had to break into three lines to make it fit this page. The ›...‹ indicate parts of the identifiers that I left out beyond that. There are in fact four map lines for `T1`-encoded Libre Caslon italics: those for lining figures ›`lf`‹, oldstyle figures ›`osf`‹, tabular lining figures ›`tlf`‹, and tabular oldstyle figures ›`tosf`‹.

The slant operation I want to add to the ⟨*special*⟩ part has the format:

```
⟨slant⟩ SlantFont
```

so for a shear to the left, for example, ⟨*slant*⟩ = −.12, which means the font gets slanted by −8° the ⟨*special*⟩ part becomes

```
"␣-0.12␣SlantFont␣AutoEnc...␣ReEncodeFont␣"
```

Finally, I select the modified font e. g. with macro

```
\usefont{⟨encoding⟩}{⟨family⟩}{⟨series⟩}{⟨shape⟩}
```

See Ref. 8 for details. In our case the call to `\usefont` is

```
\usefont{T1}{LibreCsln-OsF}{regular}{it}
```

Example

Here is all the talk of above put into action as this very document contains exactly the `\pdfmapline` just described.

Uncorrected, original italics	*White Handgloves*
Less angled version	*White Handgloves*

The only trick I have to reveal is that for the »original italics« I used the lining figures ›`lf`‹ version of the font, whereas the »less angled« version shows the oldstyle figures ›`osf`‹ version.

The TEX Font Metrics file (TFM) for this particular variant of Libre Caslon was not touched. ■

6.2 Combining LATEX and dvipdfmx

The alternative when using LATEX is similar the one elaborated in the previous section. The font mapline gets modified by `\special` primitive

```
\special{pdf:mapline ⟨font-mapline⟩}
```

that forwards the task of re-mapping the font, e. g. to **dvipdfmx**. Our running example becomes

```
\special{pdf:mapline
    LibreCsln-Italic-osf-t1-base LibreCsln-Italic
    "␣-0.12␣SlantFont␣AutoEnc...␣ReEncodeFont␣"
              <[lcsln....enc <LibreCsln-Italic.pfb}
```

The mapline contains **dvips** options for special font effects; see Ref. 10, Sec. 6.3. Note that there is no ›=‹-sign at the beginning of the `pdf:mapline` in contrast to `\pdfmapline`.

The font is activated in the same way as in the PDF-path (Sec. 6.1). The further translation of the resulting DVI-file *must* be performed with an application that is aware of the `\special` primitive as for example **dvipdfmx** [5] is.

Note

Despite the option syntax originates with **dvips** it is not able to interpret any `\special{pdf:mapline ...}`. ■

A Package Code

This is the »Reference Manual« section of the documentation where we describe the package's code and explain its implementation details.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[2005/12/01]
\ProvidesPackage{synthslant}
                [2024/12/20  v0.1a  Synthetically Slant glyphs]

\RequirePackage{etoolbox}
\RequirePackage{iftex}
\RequirePackage{xkeyval}

```

A.1 Declaration of Default Slants

`\synthslant` Introduce a reasonable default for the slant. Let the user override it if she knows better.

Remember that the slant is not an angle (with respect to the y-axis), but the sine of it; The value .2 approximately corresponds to a slant-angle of 12°.

```
\providecommand*{\synthslant}{.2}
```

`\synthnegslant` Also introduce a reasonable default for the negative slant, which is used for upright italics.

```
\providecommand*{\synthnegslant}{-.2}

```

A.2 Selection of Slant-Engine

We provide several methods to slant glyphs. The actual slanting is delegated to a ›slant-engine‹ which shears the glyphs.

`\synthslant@engine` Default to automatic selection of the slant engine.

```
\def\synthslant@engine{-1}

```

Expose default forward and backward slant values as package options.

```
\DeclareOptionX{slant}{%
  \xdef\synthslant{\fpeval{#1}}%
  \xdef\synthnegslant{\fpeval{-(#1)}}}
\DeclareOptionX{negslant}{\xdef\synthnegslant{\fpeval{#1}}}
\DeclareOptionX{posslant}{\xdef\synthslant{\fpeval{#1}}}

```

Make slant-engine selection configurable.

```
\DeclareOptionX{auto}{\def\synthslant@engine{-1}}
\DeclareOptionX{PDF}{\def\synthslant@engine{0}}
\DeclareOptionX{pdf}{\def\synthslant@engine{0}}
\DeclareOptionX{l3draw}{\def\synthslant@engine{1}}
```

```
\DeclareOptionX{ps}{\def\synthslant@engine{2}}
\DeclareOptionX{PS}{\def\synthslant@engine{2}}
\DeclareOptionX{tikz}{\def\synthslant@engine{3}}
\DeclareOptionX{TikZ}{\def\synthslant@engine{3}}
\DeclareOptionX{fontspec}{\def\synthslant@engine{4}}
\DeclareOptionX{disable}{\def\synthslant@engine{10000}}

\ProcessOptionsX\relax

```

Require sane parameter values.

```
\ExplSyntaxOn
\fp_compare:nNnTF {\synthslant} < {.0}
  {\PackageError{synthslant}
                {\string\synthslant\space <\space 0}
                {Pass\space a\space value\space that\space
                 is\space nonnegative.}}
  {}
\fp_compare:nNnTF {\synthnegslant} > {.0}
  {\PackageError{synthslant}
                {\string\synthnegslant\space >\space 0}
                {Pass\space a\space value\space that\space
                 is\space nonpositive.}}
  {}
\ExplSyntaxOff

```

Announce the positive and negative slant values now that we are sure they are ok. This may be useful information if the user passed a (complicated) floating-point expression and wants to know how LaTeX did evaluate it.

```
\PackageInfo{synthslant}{\string\synthslant=\synthslant}
\PackageInfo{synthslant}{\string\synthnegslant=\synthnegslant}


```

A.3 Slant Engines

The auto-selection code is pretty trivial. If we identify pdfTeX running we select the PDF-engine, for LuaLaTeX we select the fontspec-engine, and in all other cases we let the l3draw-layer handle the shearing.

```
\ifnum\synthslant@engine<0
  \PackageInfo{synthslant}{auto-selecting slant engine}

  \ifpdftex
    \ifnum\pdfoutput>0
      \def\synthslant@engine{0}
    \else
      \def\synthslant@engine{1}
    \fi
  \else
    \ifluatex
```

```
      \def\synthslant@engine{4}
    \else
      \def\synthslant@engine{1}
    \fi
  \fi
\fi
```

`\synthslant@shear@box` The various slant engine macros are all subsumed under `\synthslant@shear@box`. So the higher-level code becomes (almost) engine independent.

`\synthslant@engine@name` Sometimes we would like to recover the (printable) name of the selected slant engine.

```
\newcommand*{\synthslant@engine@name}{%
  \ifcase\synthslant@engine
    PDF%
  \or% 1
    l3draw%
  \or% 2
    PSTricks%
  \or% 3
    TikZ%
  \or% 4
    fontspec%
  \else
    null-implementation%
  \fi
}
```

A.3.1 PDF Slant Engine

The PDF-engine works well and it is the best tested alternative.

```
\ifcase\synthslant@engine%  0: PDF
  \PackageInfo{synthslant}{shearing done by PDF}

  \newbox{\synthslant@box}
```

`\synthslant@pdf@shear@box`

```
  \newcommand*{\synthslant@pdf@shear@box}[2]{%
    \mbox{\sbox{\synthslant@box}{#2}%
          \hskip\wd\synthslant@box
          \pdfsave
          \pdfsetmatrix{1 0 #1 1}%
          \llap{\usebox{\synthslant@box}}%
          \pdfrestore}%
  }

  \let\synthslant@shear@box=\synthslant@pdf@shear@box
```

A.3.2 l3draw Slant Engine

Using LaTeX3 may be like cheating on a very high level as the draw subsystem may delegate to the PDF-engine itself. LOL!

```
\or%  1: LaTeX3 draw subsystem
  \PackageInfo{synthslant}{shearing delegated to l3draw}

  \RequirePackage{l3draw}

  \ExplSyntaxOn
```

synthslant@latex@shear@box Slanting implemented with the experimental l3draw subsystem.

> ***Anticipated Change***
>
> As soon as the l3kernel offers an x-shear operation (`\box_xshear:Nn`?) we shall ditch this implementation and switch to the one that is tailored to *text* instead of the current one for graphics.

```
  \NewDocumentCommand{\synthslant@latex@shear@box}{mm}{
    \hbox_set:Nn \l_tmpa_box {#2}
    \dim_set:Nn \l_tmpa_dim {\box_wd:N \l_tmpa_box}
    \dim_set:Nn \l_tmpb_dim {\box_ht:N \l_tmpa_box}
    \draw_begin:
      \draw_transform_xslant:n {#1}
```

Force the baseline of the payload (#2) to coincide with the baseline of the surrounding text. This – of course – screws up our bounding box at least vertically.

```
      \box_set_dp:Nn \l_tmpa_box {\z@}
```

Here comes a fudge because the l3draw bounding boxes are way too loose. For positive slants: shrink the box-width by the box-height times ⟨*slant*⟩. For negative slants: shrink the box-width as for positive slants and in addition shift the payload to the left by the box-height times ⟨*slant*⟩.

```
      \fp_compare:nNnTF {#1} >= {.0}
        {
          \box_set_wd:Nn \l_tmpa_box
              {\l_tmpa_dim - #1\l_tmpb_dim}
        }
        {
          \draw_suspend_begin:
            \kern#1\l_tmpb_dim
          \draw_suspend_end:
          \box_set_wd:Nn \l_tmpa_box
              {\l_tmpa_dim + #1\l_tmpb_dim}
        }
```

Now typeset the box.

```
      \draw_box_use:N \l_tmpa_box
    \draw_end:
  }

  \ExplSyntaxOff

  \let\synthslant@shear@box=\synthslant@latex@shear@box
```

A.3.3 PSTricks Slant Engine

Shearing via PSTricks works, but exhibits a weird interface.

```
\or%  2: PSTricks
  \PackageInfo{synthslant}
              {shearing deferred to PostScript via PSTricks}

  \RequirePackage{pst-3d}% \pstilt

```

Package pstricks offers `\pstilt` and `\psTilt` both with typographically sub-optimal outcomes.

`thslant@pstricks@shear@box`

```
  \newcommand*{\synthslant@pstricks@shear@box}[2]{%
    \pstilt{\fpeval{57.2958 * acos(#1)}}{#2}%
  }

  \let\synthslant@shear@box=\synthslant@pstricks@shear@box
```

A.3.4 Ti*k*Z Slant Engine

The Ti*k*Z code has not been tested thoroughly yet, but it looks like it could work after some tweaking.

```
\or%  3: TikZ
  \PackageInfo{synthslant}{shearing by TikZ}

  \RequirePackage{tikz}

```

`\synthslant@tikz@shear@box`

```
  \newcommand*{\synthslant@tikz@shear@box}[2]{%
    \tikz[baseline = (ANCHOR.base), xslant = #1]
    \node[inner sep = 0pt, xslant = #1] (ANCHOR) {#2};
  }

  \let\synthslant@shear@box=\synthslant@tikz@shear@box
```

A.3.5 fontspec

The fontspec works particularly well, but it does not jibe with pdfTEX.

```
\or%  4: fontspec
  \PackageInfo{synthslant}
              {use fontspec's artificial font transformations}

  \RequirePackage{fontspec}

  \ExplSyntaxOn
```

antbox@fontspect@shear@box

```
\newcommand*{\synthslantbox@fontspect@shear@box}[2]{
  \begingroup
  \expandafter
  \fontspec[FakeSlant=#1]{\l_fontspec_family_tl}
  #2
  \endgroup
}

\ExplSyntaxOff

\let\synthslant@shear@box=\synthslantbox@fontspect@shear@box
```

A.3.6 Null Implementation

The null implementation – which does exactly what its name implies – can be useful for debugging or to get rid of the effect temporarily.

```
\else%  >=5: Null implementation
  \PackageWarning{synthslant}{shearing disabled}

```

thslant@identity@shear@box

```
  \newcommand*{\synthslant@identity@shear@box}[2]{#2}

  \let\synthslant@shear@box=\synthslant@identity@shear@box
\fi


```

A.4 Generic Slant Code

Here comes the engine-independent code.

\synthslant@nolinebreak The LaTeX3 and Ti*k*Z engines break lines at ›unexpected‹ points. Here is a duct-tape solution for them that concretes together the adajacent parts.

```
\def\synthslant@nolinebreak{%
  \ifnum\synthslant@engine=1% l3draw
    \nolinebreak
  \else
    \ifnum\synthslant@engine=3% TikZ
      \nolinebreak
    \fi
  \fi
}

```

\synthslantbox@soft@hyphen Allow for line breaks at hyphenation opportunities (›\-‹).

```
\def\synthslantbox@soft@hyphen#1\-#2\relax{%
  \synthslant@shear@box{\synthslant@slant@value}{#1}%
  \ifx\relax#2%
```

```
    \relax
  \else
    \synthslant@nolinebreak
    \discretionary{-}{}{}%
    \synthslantbox@soft@hyphen#2\relax
  \fi
}
```

`\synthslantbox@hard@hyphen` Allow for line breaks at embedded, explicit hyphens (›-‹).

```
\def\synthslantbox@hard@hyphen#1-#2\relax{%
  \synthslantbox@soft@hyphen#1\-\relax
  \ifx\relax#2%
    \relax
  \else
    \synthslant@nolinebreak
    \synthslant@shear@box{\synthslant@slant@value}{-}%
    \synthslant@nolinebreak
    \discretionary{}{}{}%
    \synthslantbox@hard@hyphen#2\relax
  \fi
}
```

`\synthslantbox@space` Allow for line breaks at embedded spaces (›␣‹).

```
\def\synthslantbox@space#1 #2\relax{%
  \synthslantbox@hard@hyphen#1-\relax
  \ifx\relax#2%
    \relax
  \else
    \space
    \synthslantbox@space#2\relax
  \fi
}
```

`\synthslantbox` We define two completely different implementations depending on the request for fontspec doing the slanting or any other package.

Macro 1: Immediately call the fontspec-specific macro. Bypass the hierarchy needed for the other slant engines.

```
\ifnum\synthslant@engine=4% fontspec
  \newrobustcmd*{\synthslantbox}[2]{%
```

The following (expanding) definition is only here for the compatibility of both branches.

```
    \edef\synthslant@slant@value{#1}
    \synthslantbox@fontspect@shear@box{\synthslant@slant@value}
                                      {#2}%
  }
```

Macro 2: This is the firestarter for the processing of all different kinds breakpoints until we reach unbreakable chunks to be passed on to the selected slant engine.

Normally, a user wants to call `\textsynthslant` or `\textsynthupright-italic`, however LaTeX wizards may have other ideas.

```
\else
  \newrobustcmd*{\synthslantbox}[2]{%
    \edef\synthslant@slant@value{#1}%
    \expandafter\synthslantbox@space#2 \relax\relax
  }
\fi

```

`box@right@slant@correction` This is a simple yet surprisingly effective heuristic for slant correction on the right-hand side if the slanted text. The value `\synthslant` is $\sin\alpha$, where α is the slant angle; see Equ. 1 on p. 3. Multiplied with the ex-height of the current font, `\fontdimen5`, this is a good approximation of the necessary slant correction.

```
\newcommand*{\synthslantbox@right@slant@correction}{%
  \dimen0=\fontdimen5\font
  \kern\synthslant\dimen0\relax
}

```

`\slantcontext` If we have microtype support we enter the context defined by this macro in `slantenvironment`.

```
\newcommand*{\slantcontext}{tracking=synthslant}

```

`slantenvironment` (*env.*) We use this environment as a pair of hooks that are called right before and right after the actual slanting code runs. The default sets up an upright type shape before and adds some italic correction after slanting.

```
\NewDocumentEnvironment{slantenvironment}{}
  {\upshape
   \ifcsdef{microtypecontext}
           {\expandafter\microtypecontext
            \expandafter{\slantcontext}}
           {}}
  {\ifcsdef{endmicrotypecontext}
           {\endmicrotypecontext}
           {}%
   \synthslantbox@right@slant@correction}

```

`\textsynthslant` User-level macro to slant some text.

```
\NewDocumentCommand{\textsynthslant}{m}
  {\ifmmode
     \synthslantbox{\synthslant}{#1}%
   \else
     {\slantenvironment
      \synthslantbox{\synthslant}{#1}%
```

```
      \endslantenvironment}%
  \fi}
```

`@right@negslant@correction` We could play the same trick here as in `\synthslantbox@right@slant@correction` and use `\synthnegslant` instead of `\synthslant`. But my experiments show no need for a correction. Anyhow, this macro may be convenient to override someday.

```
\newcommand*{\synthslantbox@right@negslant@correction}{}
```

`\negslantcontext` If we have microtype support we enter the context defined by this macro in `negslantenvironment`.

```
\newcommand*{\negslantcontext}{tracking=synthnegslant}
```

`negslantenvironment` (*env.*) We use this environment as a pair of hooks that are called right before and right after the actual un-slanting code runs.

The default sets up an italics shape before un-slanting and adds some negative italic correction after un-slanting.

```
\NewDocumentEnvironment{negslantenvironment}{}
  {\itshape
   \ifcsdef{microtypecontext}
           {\expandafter\microtypecontext
            \expandafter{\negslantcontext}}
           {}}
  {\ifcsdef{endmicrotypecontext}
           {\endmicrotypecontext}
           {}%
   \synthslantbox@right@negslant@correction}
```

`\textsynthuprightitalic` User-level macro to un-slant some italics or oblique text.

```
\NewDocumentCommand{\textsynthuprightitalic}{m}
  {\ifmmode
     \synthslantbox{\synthnegslant}{#1}%
   \else
     {\negslantenvironment
      \synthslantbox{\synthnegslant}{#1}%
      \endnegslantenvironment}%
   \fi}
```

Change History

v0.1
: General: Initial version. i

v0.1a
: General: Add missing dependency on etoolbox. Fix suggested by `mbertucci47`. 18

References

[1] Arseneau, Donald. *The relsize package*. 2013, https://ctan.org/pkg/relsize.

[2] Bienz, Tim and Richard Cohn. *Portable Document Format Reference Manual*. Addison-Wesley Publishing Company, Reading/MA, 1993, https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/pdfreference1.0.pdf.

[3] Bezos, Javier. *Package babel*. 2021, https://ctan.org/pkg/babel. The original author of babel was J. L. Braams.

[4] Carlisle, David. *Shear Transform a Box*. 2013-12-7, https://tex.stackexchange.com/questions/63179/shear-transform-a-box/63188.

[5] Dvipdfmx Project Team, ed. ***dvipdfmx***. 2020, https://ctan.org/pkg/dvipdfmx.

[6] LaTeX3 Project Team, ed. *LaTeX 2ε font selection*. 2023, https://www.latex-project.org/help/documentation/fntguide.pdf.

[7] LaTeX3 Project, *The LaTeX3 Interfaces*. 2024, https://texdoc.org/serve/interface3/0.

[8] latexref.xyz. *LaTeX 2ε: An unofficial reference manual*. 2023, https://latexref.xyz/dev/latex2e.pdf.

[9] Middendorp, Jan. *Shaping Text*. bis publishers, Amsterdam, 2014.

[10] Rokicki, Tomas. ***dvips***. 2022, https://tug.org/texlive/Contents/live/texmf-dist/doc/dvips/dvips.pdf.

[11] Schlicht, Robert. *Package microtype*. 2020, https://ctan.org/pkg/microtype.

[12] Schröder, Martin. *pdftex 1.40*. 2007, https://tug.org/mail-archives/pdftex/2007-January/006910.html.

[13] Stamm, Philipp. *Schrifttypen – Verstehen Kombinieren: Schriftmischung als Reiz in der Typografie*. Birkhäuser, Basel, 2020.

[14] Strizver, Ilene. *Type rules!: the designer's guide to professional typography*, 4th ed. John Wiley & Sons, Hoboken/NJ, 2014.

[15] Thành, Han The. *The pdfTeX user manual*. Baskerville, 8(1), 9–14 (1998), http://uk-tug-archive.tug.org/wp-installed-content/uploads/2008/12/81.pdf.

[16] Thành, Han The et al. *pdfTeX*. 2023, http://mirrors.ctan.org/systems/doc/pdftex/manual/pdftex-a.pdf.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used. We prefix all references to code lines with ›ℓ‹.