

SynthSlant – Synthetically Slanted Glyphs

Ch. L. Spiel*

v0.1 2024/03/16

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.

fga

fga *fga*

fga

This package is copyright © 2024 Ch. L. Spiel. It may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3c of this license or (at your option) any later version. This work has the LPPL maintenance status »author-maintained«.

* cspiel@users.sourceforge.org

Contents

1	Introduction	1
1.1	Appeal for Artificially Slanted Type	1
1.2	Some History	2
1.3	Shear Transformation, Slant, and Angle	3
1.4	Usage Ideas	4
2	Package Options	6
3	Macros and Environments	8
3.1	Variable-Like Macros	8
3.2	Basic Interface	9
3.3	Advanced Interface	10
4	Determining Slant	13
4.1	Direct Measurement	13
4.2	Comparison of Shapes	13
4.3	Exploring Further	14
5	Limitations and Known Problems	15
6	Alternative Solution	16
6.1	Using pdfTeX	16
6.2	Combining L ^A T _E X and dvipdfmx	17
	Changes	19
	Bibliography	20
	Index	21

The font samples »fga« on the title page were generated with the help of METAPOST using »URW Palladio L« 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

1	Variable fonts with slant-axis	2
2	Fonts with slanted series	3
3	Suggested slant values	8

List of Figures

1	Shear transform	4
2	Compare slant angles	14

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 bad reputation. Whenever there is an order to round up the usual font suspects synthetically slanted, bolded¹, 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 `s i z e` axis as well as an `o p s z` 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 it creates less contrast than a proper italic [12, 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 `ä` becomes single-storey, the start of the loop of `g` moves from the far left into the middle. Alongside, the aspect ratio of both of the counters change. These defy the common guidelines [11, Ch. 6] of font pairing. We can make sense of the seeming contradiction by recog-

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

² See for example Ref. 12, p. 97, but compare p. 142 and also Ref. 7, p. 68n, for a more nuanced assessment.

nizing 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:

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, 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 [7, p. 68n]. `Synthslant` closely follows on their steps.

Some fonts in current L^AT_EX 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 T_EX, **Computer Modern 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. Furthermore, the L^AT_EX 2_ε font selection scheme provisions ›`sl`‹ for slanted shapes and ›`ui`‹ for upright italics [5]. The former is accompanied by the macros `\slshape` and `\textsl`.

Refer to Tab. 1 on the right for a brief list of variable fonts³ that offer a slant-axis⁴ that can be controlled with fontspec’s `Slant`⁵ 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.

It seems that the original idea of automatically shearing text in L^AT_EX to simulate a slanted shape goes back to DAVID CARLISLE who suggested to use the pdf_TE_X-primitive `\pdfliteral` for shearing [3]. Shortly thereafter BRUNO LE FLOCH pointed to another

TABLE 1: Some variable fonts with a slant axis (`slnt`).

Font Name
Cairo
Commissioner
Geologica
Gluten
Inter
Recursive
Roboto Flex

³ See also the L^AT_EX 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](#).

⁴ The registered axis is called `slnt` and it is not to be confused with the `ital` axis.

⁵ Since fontspec version 2.9a as of 2024/2/13.

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 Name	Font Name
Arvo	Cabin
Computer Modern 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`.

pdfTeX-primitive, namely `\pdfsetmatrix`, available with (in 2013) more recent pdfTeX versions [14].⁶ With the help of the latter affine transformations of arbitrary 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, TikZ, 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

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

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

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

⁶ The user-level manipulation of the transformation matrix has been part of the PDF-standard since its initial publication in 1993 [1, 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 [13] and the primitive `\pdfsetmatrix` joined 2007 in pdfTeX version 1.40.0 [10].

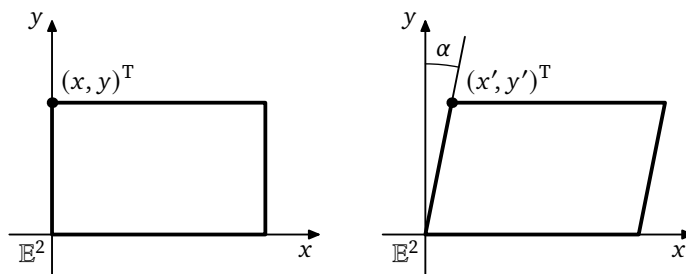


FIGURE 1: Shear transform of a rectangle by the angle α . The left-hand side shows the original figure the right-hand side the one sheared by α . The x -axis can be identified with the baseline of the text.

back and forth to the shear angle α .⁷ Some values for orientation: $\sin 5.74^\circ \approx .1$, $\sin 11.5^\circ \approx .2$, and $\sin 17.5^\circ \approx .3$.⁸ 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 L^AT_EX.

1.4 Usage Ideas

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

1. In the unfortunate situation when a serif font comes without italics such as `»URW Antiqua«`, `syntslant` can generate a slanted serif.

The user is relatively free in choosing the $\langle slant \rangle$, for there are not italics whose angle has to be matched. The package's default of .2 should be a good starting point.

2. A serif font with an italics shape can be augmented by generating upright italics.

In many cases it is desirable not to remove all forward-leaningness but retain 1° to 2° of residual slant angle.

3. 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.

⁷ At least one slant engine currently requires such a conversion, namely PSTricks. The math is hidden from the user, though.

⁸ For small angles $|\alpha|$ measured in radians the sine is approximately linear: $\sin \alpha \approx \alpha$.

4. If a sans-serif font comes without an oblique shape as, e. g., ›URW Grotesk‹, `syntslant` can generate an oblique sans-serif.
5. For sans-serif fonts with designed, this is *true* obliques as e. g. ›Open Sans‹, `syntslant` can supply a slanted sans-serif shape.
6. Fixed-width – also called ›typewriter‹ or ›teletype‹ – fonts without obliques (Yes, I am looking at you, Inconsolata!) finally get an oblique shape.
7. Small caps without accompanying italics can be slanted, too.
8. As `synthslant` works in $\text{T}_{\text{E}}\text{X}$'s math-mode, too, it is possible to give math-italics even more of a heeling.

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.

auto

Let the package choose a slant engine. This is the default.

For pdfL^AT_EX package synthslant selects the PDF-engine, for LuaL^AT_EX 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.⁹

l3draw

Select the ›draw‹ layer of L^AT_EX3 as base for the slanting engine.¹⁰

Caution

This engine is experimental and the ›draw‹ layer of L^AT_EX3 itself is still experimental, too. See Sec. 5 on p. 15 for details. ■

negslant=*<slant>*

Set the default value for `\synthnegslant` only. Note that for this option *<slant>* is a non-positive value.

PDF, pdf

Select the PDF-slant engine. This requires the document is translated with pdfL^AT_EX or a compatible program.

posslant=*<slant>*

Set the default value for `\synthslant` only. Note that for this option *<slant>* is a nonnegative value.

PS, ps

Use PSTricks to delegate slanting to the PostScript interpreter. Obviously requires PSTricks¹¹ and DVI-to-PostScript translation.

⁹ Requires *fontspec.sty*.

¹⁰ This option requires *l3draw.sty*.

¹¹ 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>*

Set the default values for both `\synthslant` and `\synthnegslant`, where `\synthslant` becomes *<slant>* and `\synthnegslant` becomes $-\langle slant \rangle$.

If omitted `\synthslant` assumes .2 and $-.2$, respectively.

TikZ, tikz

Use TikZ for slanting.¹²

Caution

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

¹² Requires *tikz.sty*.

3 Macros and Environments

This section describes how to actually apply the functionality of `synthslant` to some text. If the $\langle slant \rangle$ value matching a given font is known this is about it. To figure out an unknown $\langle slant \rangle$ 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 $\langle slant \rangle$ 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 Name	Slant	Font Name	Slant	Font Name	Slant
ADF Accanthis	.26	Day Roman	.2	Merriweather	.14
ADF Baskervald	.32	EB Garamond	.3	MLModern	.23
ADF Berenis	.2	etbb	.2	Noto Serif	.22
ADF Venturis	.2	fbb	.2	PT Serif	.2
Alegreya	.2	Garamond Expert	.2	Roboto Slab	.2
Arvo	.2	Gandhi Serif	.2	Quattrocento	.2
BaskervilleF	.2	Gentium	.2	Source Serif Pro	.18
Bera Serif	.2	Ibarra Real Nova	.2	Spectral	.18
Bitter	.16	IBM Plex Serif	.24	STIX	.2
Caladea	.14	INRIA Serif	.2	T _E X Gyre Pagella	.16
Clara	.24	Libertinus Serif	.2	TX Fonts Serif	.2
Cochineal	.2	Libre Baskerville	.3	URW Antiqua	.2
Coelacanth	.2	Libre Caslon	.38	URW Nimbus Roman	.2
Crimson Pro	.2			Utopia	.2
Crimson Text	.2				

`\synthnegslant` Control the slant applied by `\textsynthuprightitalic`. This value is non-positive.

```
\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 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`.

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` [2]. ¶

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`.

Use Case

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.

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` Wrapper around `\synthslantbox` that is called for every forward-slanting operation with `\textsynthslant`.

```
slantenvironment
```

Switch to an upright font shape and – if package `microtype` [9] 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` Wrapper around `\synthslantbox` that is called for every backward-slanting operation with `\textsynthuprightitalic`.

```
negslantenvironment
```

Switch to an italics font shape and – if package `microtype` [9] has been loaded – enter the Microtype-context defined by macro `\negslantcontext`.

`\slantcontext` 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` Microtype context used when typesetting un-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 $\langle slant \rangle$.

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 as 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. $\rangle sample \langle$.
6. On layer $\rangle sample \langle$ 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.

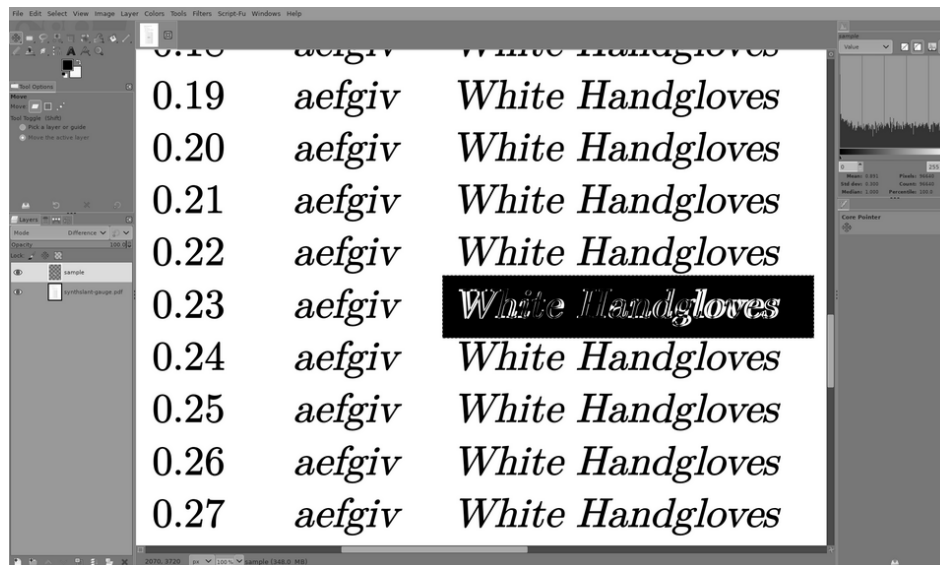


FIGURE 2: Compare italics and slanted samples with The Gimp. ¶
 For this screendump I loaded the samples on the first page of *synthslant-gauge.pdf* at a resolution of 600 dpi into The Gimp. The ›sample‹ layer is aligned to the letter ›t‹ in the word ›White‹. Note that accidentally the letter ›H‹ of 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 L^AT_EX-engine – used to examine the details of the slant operations.

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

Page 3, Sec. 3.2 and following sub-sections, examines the coupling of synthslant with the T_EX-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 loses 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.

`TikZ` engine. The `TikZ` 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 [14, 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` [8, 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 3 and generate less-angled italics for Libre Caslon. Here is a suitable map line taken from `pdfTeX.map` on my system:

```
LibreCsln-Italic-osf-tl-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 `>osf<`.

The slant operation I want to add to the $\langle special \rangle$ part has the format:

$\langle slant \rangle$ SlantFont

so for a shear to the left, for example, $\langle slant \rangle = -.12$, which means the font gets slanted by -8° the $\langle special \rangle$ part becomes

`"_-.12_SlantFont_AutoEnc..._ReEncodeFont_"`

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

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

See Ref. 6 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	<i>White Handgloves</i>
Less angled version	<i>White Handgloves</i>

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

6.2 Combining L^AT_EX and dvipdfmx

The alternative when using L^AT_EX 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
  "_-.12_SlantFont_AutoEnc..._ReEncodeFont_"
  <[lcsln....enc <LibreCsln-Italic.pfb}
```

The mapline contains **dvips** options for special font effects; see Ref. 8, 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** [4] is.

Note

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

Change History

v0.1

General: Initial version [i](#)

References

- [1] 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>.
- [2] BEZOS, JAVIER. *Package babel*. 2021, <https://ctan.org/pkg/babel>. The original author of babel was J. L. BRAAMS.
- [3] CARLISLE, DAVID. *Shear Transform a Box*. 2013-12-7, <https://tex.stackexchange.com/questions/63179/shear-transform-a-box/63188>.
- [4] DVIPDFMX PROJECT TEAM, ed. *dvipdfmx*. 2020, <https://ctan.org/pkg/dvipdfmx>.
- [5] L^AT_EX3 PROJECT TEAM, ed. *L^AT_EX 2_ε font selection*. 2023, <https://www.latex-project.org/help/documentation/fntguide.pdf>.
- [6] LATEXREF.XYZ. *L^AT_EX 2_ε: An unofficial reference manual*. 2023, <https://latexref.xyz/dev/latex2e.pdf>.
- [7] MIDDENDORP, JAN. *Shaping Text*. BIS publishers, Amsterdam, 2014.
- [8] ROKICKI, TOMAS. *dvips*. 2022, <https://tug.org/texlive/Contents/live/texmf-dist/doc/dvips/dvips.pdf>.
- [9] SCHLICHT, ROBERT. *Package microtype*. 2020, <https://ctan.org/pkg/microtype>.
- [10] SCHRÖDER, MARTIN. *pdftex 1.40*. 2007, <https://tug.org/mail-archives/pdftex/2007-January/006910.html>.
- [11] STAMM, PHILIPP. *Schrifttypen – Verstehen Kombinieren: Schriftmischung als Reiz in der Typografie*. Birkhäuser, Basel, 2020.
- [12] STRIZVER, ILENE. *Type rules!: the designer’s guide to professional typography*, 4th ed. John Wiley & Sons, Hoboken/NJ, 2014.
- [13] THÀNH, HAN THE. *The pdf_TE_X user manual*. Baskerville, 8(1), 9–14 (1998), <http://uk-tug-archive.tug.org/wp-installed-content/uploads/2008/12/81.pdf>.
- [14] THÀNH, HAN THE et al. *pdf_TE_X*. 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.

B

breakpoint (space), 9

D

discretionary hyphen, 9

E

environments:

 negslantenvironment, [11](#)

 slantenvironment, [11](#)

F

font

 ADF Accanthis, 8

 ADF Baskervald, 8

 ADF Berenis, 8

 ADF Venturis, 8

 Alegreya, 8

 Arvo, 3, 8

 axis

 ital, 2

 opsize, 1

 size, 1

 slant, 2

 BaskervilleF, 8

 Bera Serif, 8

 Bitter, 8

 Cabin, 3

 Cairo, 2

 Caladea, 8

 Clara, 8

 Clear Sans, 3

 Cochineal, 8

 Coelacanth, 8

 Commissioner, 2

 Computer Modern Roman, 2, 3

 Crimson Pro, 8

 Crimson Text, 8

 Cuprum, 3

 Day Roman, 8

 Domitian, 3

 Droid Serif, 3

 EB Garamond, 8

 Erewhon, 3

 etbb, 8

 Extended Charter, 3

 fbb, 8

 Fira Sans, 3

 Gandhi Sans, 3

 Gandhi Serif, 8

 Garamond Expert, 8

 Gentium, 8

 Geologica, 2

 GFS Artemisia, 3

 GFS Bodoni, 3

 GFS Didot, 3

 Gluten, 2

 Ibarra Real Nova, 8

 IBM Plex Serif, 8

 Inconsolata, 5

 INRIA Sans, 3

 INRIA Serif, 8

 Inter, 2

 Lato, 3

 Libertinus Serif, 8

 Libre Baskerville, 8

 Libre Caslon, 4, 8, 16

 Merriweather, 8

 MLModern, 8

 Montserrat, 3

 Noto Serif, 8

 Open Sans, 5

 PT Sans, 3

 PT Serif, 8

 Quattrocento, 8

 Recursive, 2

 Roboto Flex, 2

 Roboto Slab, 8

 Source Sans Pro, 3

 Source Serif Pro, 8

 Spectral, 8

 STIX, 8

 TeX Gyre Pagella, 8

 TX Fonts Serif, 8

URW Antiqua, 4, 8
URW Grotesk, 5
URW Nimbus Roman, 8
Utopia, 8

N

`\negslantcontext`, [11](#)
`negslantenvironment` (environment), [11](#)

P

package
 TikZ, 7
 babel, 10
 fontspec, 2, 6, 9
 l3draw, 3, 6, 15
 microtype, 11
 pstricks, 6, 15
 tikz, 15
package option
 auto, 6
 disable, 6
 fontspec, 6
 l3draw, 6

`negslant`, 6
`pdf`, 6
`posslant`, 6
`ps`, 6
`slant`, 7
`tikz`, 7

PostScript, 6

S

shear
 angle, 4
 transformation, 3
slant
 correction, 9
 operation, 3
`\slantcontext`, [11](#)
`slantenvironment` (environment), [11](#)
`\synthnegslant`, [9](#)
`\synthslant`, [8](#)
`\synthslantbox`, [10](#)

T

`\textsynthslant`, [9](#)
`\textsynthuprightitalic`, [10](#)