

SynthSlant – Synthetically Slanted Glyphs

Ch. L. Spiel*

v0.1a 2024/07/08

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](#) (LPPL), 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

Quick Reference [v](#)

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 pdf \TeX [16](#)
- 6.2 Combining L \TeX and dvipdfmx [17](#)

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.

A

Package Code 18

- A.1 Declaration of Default Slants 18
- A.2 Selection of Slant-Engine 18
- A.3 Slant Engines 19
 - A.3.1 PDF Slant Engine 20
 - A.3.2 l3draw Slant Engine 21
 - A.3.3 PSTricks Slant Engine 22
 - A.3.4 TikZ Slant Engine 22
 - A.3.5 fontspec 23
 - A.3.6 Null Implementation 23
- A.4 Generic Slant Code 23

Change History 27

References 28

Index 29

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 3
- 2 Compare slant angles 14

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 (p. 11)

`negslantenvironment`

Wrapper around `\synthslantbox` when slanting backward with `\textsynthuprightitalic` (p. 11)

`\slantcontext`

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

`slantenvironment`

Wrapper around `\synthslantbox` when slanting forward with `\textsynthslant` (p. 10)

`\synthnegslant`

Slant value used by `\textsynthuprightitalic` (p. 8)

`\synthslantbox`{*slant*}{*text*}

Slant *text* (forward or backward) with *slant* (p. 10)

`\synthslant`

Slant value used by `\textsynthslant` (p. 8)

`\textsynthslant`{*text*}

Forward slant upright glyphs (p. 9)

`\textsynthuprightitalic`{*text*}

Backward slant italics or oblique glyphs (p. 10)

1 Introduction

The `synthslant` package provides a translator (e. g. $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, $\text{pdfL}^{\text{A}}\text{T}_{\text{E}}\text{X}$, or $\text{LuaL}^{\text{A}}\text{T}_{\text{E}}\text{X}$) 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, $\text{pdfT}_{\text{E}}\text{X}$ performs a similar operation under the hood: of the 40,210 map lines in my `pdftex.map` currently 1,236 instruct $\text{pdfT}_{\text{E}}\text{X}$ 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, 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 `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 it 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 into 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 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, [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. Furthermore, the L^AT_EX 2_ε font selection scheme provisions ›`s l`‹ for slanted shapes and ›`u i`‹ 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 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 [4]. Shortly thereafter BRUNO LE FLOCH pointed to another pdf_TE_X-primitive, namely `\pdfsetmatrix`, available with (in 2013) more recent pdf_TE_X versions [16].⁶ With the help of the latter affine transformations of arbitrary content can be coded directly by setting the transformation matrix. A slight

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

Font
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.

⁶ 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 pdf_TE_X in 1998 [15] and the primitive `\pdfsetmatrix` joined 2007 in pdf_TE_X 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`.

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.

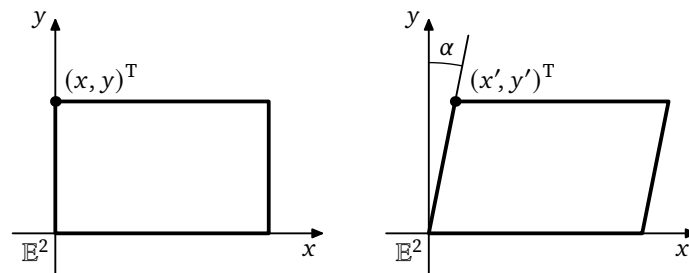


FIGURE 1: Shear transform of a rectangle by the angle α . The left-hand side shows the original figure the right-hand side the one sheared by α . The x-axis can be identified with the baseline of the text.

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

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. 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.
2. 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.
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.
4. Generate an oblique sans-serif if a sans-serif font comes without an oblique shape as, e. g., `>URW Grotesk<`.
5. Supply a slanted sans-serif shape for sans-serif fonts with designed, this is *true* obliques as e. g. `>Open Sans<`.
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` also works in T_EX's math-mode, it is possible to give math-italics even more of a heeling.
9. 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.
10. Big mathematical operators like the sigma can be slanted and others, like the integral sign, can have their inclination adjusted.

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: $\sin \alpha \approx \alpha$.

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-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 requires the document is translated with pdfL^AT_EX 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 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-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.

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. ■

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.

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	Slant	Font	Slant	Font	Slant
ADF Accanthis	.26	Day Roman	.2	Merriweather	.14
ADF Baskervald	.32	EB Garamond	.3	ML Modern	.23
ADF Berenis	.2	etbb	.2	Noto Serif	.22
ADF Venturis	.2	fbf	.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 \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 $\langle text \rangle$. 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 $\rangle\langle$ get propagated. So, we could improve on our above example by saying

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

to \rangle recover \langle 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{\langle text \rangle}
```

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

In math mode just slant $\langle text \rangle$ 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 \rangle et. al. \langle and \rangle etc. \langle :

```
\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 $\langle text \rangle$ with the slant value stored in `\synthnegslant`.

In math mode just un-slant $\langle text \rangle$ 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 $\langle text \rangle$ with an amount of $\langle slant \rangle$ 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 T_EX-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 `resize` 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. 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. $\rangle sample \langle$.
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 $\rangle sample \langle$ to $\rangle difference \langle$ 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.

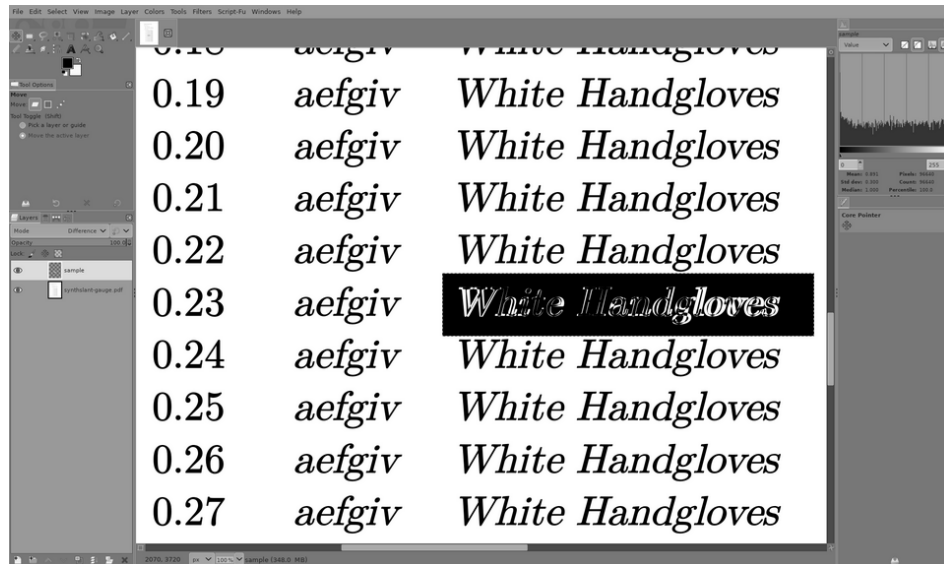


FIGURE 2: Compare italics and slanted samples with The Gimp. For this screndump I loaded the samples on the first page of *synthslant-gauge.pdf* at a resolution of 600 dpi into The Gimp. The `>sample<` layer is aligned to the letter `>t<` in the word `>White<`. Note that accidentally the letter `>H<` 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 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 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 [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 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-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 `>osf<`.

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	<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
  "_-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.

```

1 ⟨*package⟩
2 \NeedsTeXFormat{LaTeX2e}[2005/12/01]
3 \ProvidesPackage{synthslant}
4           [2024/07/08 v0.1a Synthetically Slant glyphs]
5
6 \RequirePackage{etoolbox}
7 \RequirePackage{iftex}
8 \RequirePackage{xkeyval}
9

```

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° .

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

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

```
11 \providecommand*{\synthnegslant}{-.2}
12

```

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.

```
13 \def\synthslant@engine{-1}
14

```

Expose default forward and backward slant values as package options.

```

15 \DeclareOptionX{slant}{%
16   \xdef\synthslant{\fpeval{#1}}%
17   \xdef\synthnegslant{\fpeval{-(#1)}}}
18 \DeclareOptionX{negslant}{\xdef\synthnegslant{\fpeval{#1}}}
19 \DeclareOptionX{posslant}{\xdef\synthslant{\fpeval{#1}}}
20

```

Make slant-engine selection configurable.

```

21 \DeclareOptionX{auto}{\def\synthslant@engine{-1}}
22 \DeclareOptionX{PDF}{\def\synthslant@engine{0}}
23 \DeclareOptionX{pdf}{\def\synthslant@engine{0}}
24 \DeclareOptionX{l3draw}{\def\synthslant@engine{1}}

```

```

25 \DeclareOptionX{ps}{\def\synthslant@engine{2}}
26 \DeclareOptionX{PS}{\def\synthslant@engine{2}}
27 \DeclareOptionX{tikz}{\def\synthslant@engine{3}}
28 \DeclareOptionX{TikZ}{\def\synthslant@engine{3}}
29 \DeclareOptionX{fontspec}{\def\synthslant@engine{4}}
30 \DeclareOptionX{disable}{\def\synthslant@engine{10000}}
31
32 \ProcessOptionsX\relax
33

```

Require sane parameter values.

```

34 \ExplSyntaxOn
35 \fp_compare:nNnTF {\synthslant} < {.0}
36   {\PackageError{synthslant}
37     {\string\synthslant\space <\space 0}
38     {Pass\space a\space value\space that\space
39       is\space nonnegative.}}
40   {}
41 \fp_compare:nNnTF {\synthnegslant} > {.0}
42   {\PackageError{synthslant}
43     {\string\synthnegslant\space >\space 0}
44     {Pass\space a\space value\space that\space
45       is\space nonpositive.}}
46   {}
47 \ExplSyntaxOff
48

```

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 L^AT_EX did evaluate it.

```

49 \PackageInfo{synthslant}{\string\synthslant=\synthslant}
50 \PackageInfo{synthslant}{\string\synthnegslant=\synthnegslant}
51
52

```

A.3 Slant Engines

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

```

53 \ifnum\synthslant@engine<0
54   \PackageInfo{synthslant}{auto-selecting slant engine}
55
56   \ifpdfTEX
57     \ifnum\pdfoutput>0
58       \def\synthslant@engine{0}
59     \else
60       \def\synthslant@engine{1}
61     \fi
62   \else
63     \ifluaTEX

```



```

64     \def\synthslant@engine{4}
65     \else
66         \def\synthslant@engine{1}
67     \fi
68 \fi
69 \fi
70
71

```

`\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.

```

72 \newcommand*{\synthslant@engine@name}{%
73     \ifcase\synthslant@engine
74         PDF%
75     \or% 1
76         l3draw%
77     \or% 2
78         PSTricks%
79     \or% 3
80         TikZ%
81     \or% 4
82         fontspec%
83     \else
84         null-implementation%
85     \fi
86 }
87

```

A.3.1 PDF Slant Engine

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

```

88 \ifcase\synthslant@engine% 0: PDF
89     \PackageInfo{synthslant}{shearing done by PDF}
90
91     \newbox{\synthslant@box}
92

```

`\synthslant@pdf@shear@box`

```

93     \newcommand*{\synthslant@pdf@shear@box}[2]{%
94         \mbox{\sbox{\synthslant@box}{#2}%
95             \hskip\wd\synthslant@box
96             \pdfsave
97             \pdfsetmatrix{1 0 #1 1}%
98             \llap{\usebox{\synthslant@box}}%
99             \pdfrestore}%
100     }
101
102     \let\synthslant@shear@box=\synthslant@pdf@shear@box

```

A.3.2 l3draw Slant Engine

Using L^AT_EX3 may be like cheating on a very high level as the draw subsystem may delegate to the PDF-engine itself. LOL!

```

103 \or% 1: LaTeX3 draw subsystem
104 \PackageInfo{synthslant}{shearing delegated to l3draw}
105
106 \RequirePackage{l3draw}
107
108 \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.

```

109 \NewDocumentCommand{\synthslant@latex@shear@box}{mm}{
110   \hbox_set:Nn \l_tmpa_box {#2}
111   \dim_set:Nn \l_tmpa_dim {\box_wd:N \l_tmpa_box}
112   \dim_set:Nn \l_tmpb_dim {\box_ht:N \l_tmpa_box}
113   \draw_begin:
114     \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.

```

115     \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>*.

```

116     \fp_compare:nNnTF {#1} >= {.0}
117     {
118       \box_set_wd:Nn \l_tmpa_box
119         {\l_tmpa_dim - #1\l_tmpb_dim}
120     }
121     {
122       \draw_suspend_begin:
123         \kern#1\l_tmpb_dim
124       \draw_suspend_end:
125       \box_set_wd:Nn \l_tmpa_box
126         {\l_tmpa_dim + #1\l_tmpb_dim}
127     }

```

Now typeset the box.

```

128     \draw_box_use:N \l_tmpa_box
129   \draw_end:
130 }

```

```

131 \ExplSyntaxOff
132
133 \let\synthslant@shear@box=\synthslant@latex@shear@box

```

A.3.3 PSTricks Slant Engine

Shearing via PSTricks works, but exhibits a weird interface.

```

134 \or% 2: PSTricks
135 \PackageInfo{synthslant}
136           {shearing deferred to PostScript via PSTricks}
137
138 \RequirePackage{pst-3d}% \pstilt
139

```

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

```
\synthslant@pstricks@shear@box
```

```

140 \newcommand*{\synthslant@pstricks@shear@box}[2]{%
141   \pstilt{\fpeval{57.2958 * acos(#1)}}{#2}%
142 }
143
144 \let\synthslant@shear@box=\synthslant@pstricks@shear@box

```

A.3.4 TikZ Slant Engine

The TikZ code has not been tested thoroughly yet, but it looks like it could work after some tweaking.

```

145 \or% 3: TikZ
146 \PackageInfo{synthslant}{shearing by TikZ}
147
148 \RequirePackage{tikz}
149

```

```
\synthslant@tikz@shear@box
```

```

150 \newcommand*{\synthslant@tikz@shear@box}[2]{%
151   \tikz[baseline = (ANCHOR.base), xslant = #1]
152   \node[inner sep = 0pt, xslant = #1] (ANCHOR) {#2};
153 }
154
155 \let\synthslant@shear@box=\synthslant@tikz@shear@box

```

A.3.5 fontspec

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

```

156 \or% 4: fontspec
157 \PackageInfo{synthslant}
158           {use fontspec's artificial font transformations}
159
160 \RequirePackage{fontspec}
161
162 \ExplSyntaxOn

```

antbox@fontspect@shear@box

```

163 \newcommand*{\synthslantbox@fontspect@shear@box}[2]{
164   \begingroup
165   \expandafter
166   \fontspec[FakeSlant=#1]{\l_fontspec_family_tl}
167   #2
168   \endgroup
169 }

170 \ExplSyntaxOff
171
172 \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.

```

173 \else% >=5: Null implementation
174 \PackageWarning{synthslant}{shearing disabled}
175

```

thslant@identity@shear@box

```

176 \newcommand*{\synthslant@identity@shear@box}[2]{#2}
177

178 \let\synthslant@shear@box=\synthslant@identity@shear@box
179 \fi
180
181

```

A.4 Generic Slant Code

Here comes the engine-independent code.

`\synthslant@nolinebreak` The L^AT_EX3 and TikZ engines break lines at >unexpected< points. Here is a duct-tape solution for them that concretes together the adajacent parts.

```

182 \def\synthslant@nolinebreak{%
183   \ifnum\synthslant@engine=1% l3draw
184     \nolinebreak
185   \else

```

```

186   \ifnum\synthslant@engine=3% TikZ
187     \nolinebreak
188   \fi
189 \fi
190 }
191

```

`\synthslantbox@soft@hyphen` Allow for line breaks at hyphenation opportunities ($\>\<$).

```

192 \def\synthslantbox@soft@hyphen#1\-#2\relax{%
193   \synthslant@shear@box{\synthslant@slant@value}{#1}%
194   \ifx\relax#2%
195     \relax
196   \else
197     \synthslant@nolinebreak
198     \discretionary{-}{}{}%
199     \synthslantbox@soft@hyphen#2\relax
200   \fi
201 }
202

```

`\synthslantbox@hard@hyphen` Allow for line breaks at embedded, explicit hyphens ($\>\<$).

```

203 \def\synthslantbox@hard@hyphen#1-#2\relax{%
204   \synthslantbox@soft@hyphen#1-\relax
205   \ifx\relax#2%
206     \relax
207   \else
208     \synthslant@nolinebreak
209     \synthslant@shear@box{\synthslant@slant@value}{-}%
210     \synthslant@nolinebreak
211     \discretionary{}{}{}%
212     \synthslantbox@hard@hyphen#2\relax
213   \fi
214 }
215

```

`\synthslantbox@space` Allow for line breaks at embedded spaces ($\>_<$).

```

216 \def\synthslantbox@space#1 #2\relax{%
217   \synthslantbox@hard@hyphen#1-\relax
218   \ifx\relax#2%
219     \relax
220   \else
221     \space
222     \synthslantbox@space#2\relax
223   \fi
224 }
225

```

`\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.

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

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

```
228 \edef\synthslant@slant@value{#1}
229 \synthslantbox@fontspec@shear@box{\synthslant@slant@value}
230 {#2}%
231 }
```

Macro 2: This is the firestarter for the processing of all different kinds break-points 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 L^AT_EX wizards may have other ideas.

```
232 \else
233 \newrobustcmd*{\synthslantbox}[2]{%
234 \edef\synthslant@slant@value{#1}%
235 \expandafter\synthslantbox@space#2 \relax\relax
236 }
237 \fi
238
```

`\synthslantbox@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.

```
239 \newcommand*{\synthslantbox@right@slant@correction}{%
240 \dimen0=\fontdimen5\font
241 \kern\synthslant\dimen0\relax
242 }
243
```

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

```
244 \newcommand*{\slantcontext}{tracking=synthslant}
245
```

`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.

```
246 \NewDocumentEnvironment{slantenvironment}{}
247 {\upshape
248 \ifcsdef{microtypecontext}
249 {\expandafter\microtypecontext
250 \expandafter{\slantcontext}}
251 {}
252 {\ifcsdef{endmicrotypecontext}
253 {\endmicrotypecontext}
254 {}%
255 \synthslantbox@right@slant@correction}
256
```

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

```

257 \NewDocumentCommand{\textsynthslant}{m}
258   {\ifmmode
259     \synthslantbox{\synthslant}{#1}%
260   \else
261     {\slantenvironment
262     \synthslantbox{\synthslant}{#1}%
263     \endslantenvironment}%
264   \fi}
265

```

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

```

266 \newcommand*{\synthslantbox@right@negslant@correction}{}
267

```

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

```

268 \newcommand*{\negslantcontext}{tracking=synthnegslant}
269

```

`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.

```

270 \NewDocumentEnvironment{negslantenvironment}{}
271   {\itshape
272   \ifcsdef{microtypecontext}
273     {\expandafter\microtypecontext
274     \expandafter{\negslantcontext}}
275   {}
276   {\ifcsdef{endmicrotypecontext}
277     {\endmicrotypecontext}
278   {}%
279   \synthslantbox@right@negslant@correction}
280

```

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

```

281 \NewDocumentCommand{\textsynthuprightitalic}{m}
282   {\ifmmode
283     \synthslantbox{\synthnegslant}{#1}%
284   \else
285     {\negslantenvironment
286     \synthslantbox{\synthnegslant}{#1}%
287     \endnegslantenvironment}%
288   \fi}
289

```

Change History

v0.1

General: Initial version. [i](#)

v0.1a

General: Add missing dependency on etoolbox. Fix suggested by [mber tucci47](#). [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] L^AT_EX3 PROJECT TEAM, ed. *L^AT_EX 2 ϵ font selection*. 2023, <https://www.latex-project.org/help/documentation/fntguide.pdf>.
- [7] L^AT_EX3 PROJECT, *The L^AT_EX3 Interfaces*. 2024, <https://texdoc.org/serve/interface3/0>.
- [8] LATEXREF.XYZ. *L^AT_EX 2 ϵ : 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 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>.
- [16] 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. We prefix all references to code lines with `>ℓ<`.

A

auto (option) 6

B

breakpoint at space 9

D

disable (option) 6

discretionary hyphen 9

E

environments:

 nagslantenvironment *11*, ℓ270

 slantenvironment *10*, ℓ246

F

font

 axis

 ital 2

 opsz 1

 size 1

 slant 2

 typeface

 ADF Accanthis 8

 ADF Baskervald 8

 ADF Berenis 8

 ADF Venturis 8

 Alegreya 8

 Arvo 3, 8

 BaskervilleF 8

 Bera Serif 8

 Bitter 8

 Cabin 3

 Cairo 2

 Caladea 8

 Clara 8

 Clear Sans 3

 CM Roman 2, 3

 Cochineal 8

 Coelacanth 8

 Commissioner 2

 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 4

 INRIA Sans 3

 INRIA Serif 8

 Inter 2

 Lato 3

 Libertinus Serif 8

 Libre Baskerville 8

 Libre Caslon 4, 8, 16

 Merriweather 8

 ML Modern 8

 Montserrat 3

 Noto Serif 8

 Open Sans 4

 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

 T_EX Gyre Pagella 8

- TX Fonts Serif [8](#)
 - URW Antiqua [4](#), [8](#)
 - URW Grotesk [4](#)
 - URW Nimbus Roman [8](#)
 - Utopia [8](#)
 - fontspec (option) [6](#)
- L**
- l3draw (option) [6](#)
- N**
- negslant (option) [6](#)
 - \negslantcontext [11](#), [l268](#), [l274](#)
 - \negslantenvironment [l285](#)
 - negslantenvironment (env.) [11](#), [l270](#)
- P**
- package
 - babel [9](#)
 - fontspec [2](#), [6](#), [9](#)
 - l3draw [3](#), [6](#), [15](#)
 - microtype [11](#)
 - pstricks [6](#), [15](#)
 - tikz [7](#), [15](#)
 - package option
 - auto [6](#)
 - disable [6](#)
 - fontspec [6](#)
 - l3draw [6](#)
 - negslant [6](#)
 - pdf [6](#)
 - posslant [6](#)
 - slant [7](#)
 - tikz [7](#)
 - pdf (option) [6](#)
 - posslant (option) [6](#)
 - PostScript [6](#)
 - ps [6](#)
- S**
- shear
 - angle [4](#)
 - transformation [3](#)
 - slant
 - correction [9](#)
 - operation [3](#)
 - slant (option) [7](#)
 - \slantcontext [11](#), [l244](#), [l250](#)
 - \slantenvironment [l261](#)
 - slantenvironment (env.) [10](#), [l246](#)
 - \synthnegslant [8](#), [l11](#), [l17](#), [l18](#), [l41](#), [l43](#), [l50](#), [l283](#), [l286](#)
 - \synthslant [8](#), [l10](#), [l16](#), [l19](#), [l35](#), [l37](#), [l49](#), [l241](#), [l259](#), [l262](#)
 - \synthslantbox [10](#), [l226](#), [l259](#), [l262](#), [l283](#), [l286](#)
- T**
- \textsynthslant [9](#), [l257](#)
 - \textsynthuprightitalic [10](#), [l281](#)
 - tikz (option) [7](#)