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.

This package is copyright © 2024 Ch. L. Spiel. It may be distributed and/or modified under the conditions of the LaTeX 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*
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
1 Introduction

The synthslant package provides a translator (e. g. L\LaTeX, pdfL\LaTeX, or LuaL\LaTeX) 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, pdf\LaTeX performs a similar operation under the hood: of the 40,210 map lines in my pdf\TeX.map currently 1,236 instruct pdf\TeX 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, 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 \(\text{a}\) becomes single-storey, the start of the loop of \(\text{g}\) 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 \textbackslash pmb that works by overprinting. The authors of amsbsy recommend to prefer package \textbackslash 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.D. 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**, **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 **LaTeX** 2e 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 fonts³ that offer a slant-axis⁴ that can be controlled with **fontspec**’s Slant⁵ 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 **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 Floc’h** pointed to another **pdftex-**primitive, namely \pdfsetmatrix, available with (in 2013) more recent **pdftex** 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

---

³ 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**.

⁴ The registered axis is called slnt and it is not to be confused with the Italic 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 **pdftex** in 1998 [15] and the primitive \pdfsetmatrix joined 2007 in **pdftex** version 1.40.0 [12].
1.3 Introduction: Shear Transformation, Slant, and Angle

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

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

(1)

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

For \(\alpha = 0\) the shear matrix becomes the identity matrix. Throughout of synthslant we work with the \(<\text{slant}>\) which is \(\sin \alpha\) in Equ. 1 and avoid converting

\[\text{variant of his code is used in this package for the PDF- and l3draw-slant engines. The implementations for PSTricks, TikZ, and fontspecfontspec are trivial as they build upon shear functions supplied by the respective packages.}\]

### Table 2: Selected fonts that come with their own slanted series

<table>
<thead>
<tr>
<th>Font Name</th>
<th>Font Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arvo</td>
<td>Cabin</td>
</tr>
<tr>
<td>Computer Modern Roman</td>
<td>Clear Sans</td>
</tr>
<tr>
<td>Domitian</td>
<td>Cuprum</td>
</tr>
<tr>
<td>Droid Serif</td>
<td>Fira Sans</td>
</tr>
<tr>
<td>Erewhon</td>
<td>Gandhi Sans</td>
</tr>
<tr>
<td>Extended Charter</td>
<td>INRIA Sans</td>
</tr>
<tr>
<td>GFS Artemisia</td>
<td>Lato(^{\dagger})</td>
</tr>
<tr>
<td>GFS Bodoni</td>
<td>Montserrat</td>
</tr>
<tr>
<td>GFS Didot</td>
<td>PT Sans</td>
</tr>
<tr>
<td></td>
<td>Source Sans Pro</td>
</tr>
</tbody>
</table>

\(^{\dagger}\) The shape is activated with \texttt{\itshape}.

\[\text{FIGURE 1: Shear transform of a rectangle by the angle } \alpha. \text{ The left-hand side shows the original figure the right-hand side the one sheared by } \alpha. \text{ The } x\text{-axis can be identified with the baseline of the text.}\]
back and forth to the shear angle $\alpha$. Some values for orientation: $\sin 5.74^\circ \approx .1$, $\sin 11.5^\circ \approx .2$, and $\sin 17.5^\circ \approx .3$. For real-life serif fonts (slant) is in the range of .1 to .45 and a value of .2 seems to be quite common. My pdftex.map lists negative (slant) values in the range $-.4$ to $-.05$ and positive (slant) 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.

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 (slant), 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 $1^\circ$ to $2^\circ$ 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 \TeX'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.

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

\footnote{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

\begin{verbatim}
\usepackage[⟨option⟩⋯]{synthslant}
\end{verbatim}

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.

\texttt{auto}
Let the package choose a slant engine. This is the default.

\texttt{For pdflatex} package synthslant selects the \texttt{PDF}-engine, for \texttt{Lualatex} the \texttt{fontspec}-engine, and in all other cases the \texttt{l3draw}-layer handles the shear transformation.

\texttt{disable}
Disable slanting completely.

\texttt{fontspec}
Use \texttt{fontspec} as slanting back-end.\(^9\)

\texttt{l3draw}
Select the \texttt{draw} layer of \LaTeX3 as base for the slanting engine.\(^{10}\)

\textbf{Caution}
This engine is experimental and the \texttt{draw} layer of \LaTeX3 itself is still experimental, too. See Sec. 5 on p. 15 for details.

\texttt{negslant=⟨slant-expr⟩}
Set the default value for \texttt{\synthnegslant} only. The argument \texttt{⟨slant-expr⟩} is a floating-point expression. Note that for this option \texttt{⟨slant-expr⟩} must evaluate to a non-positive value.

\texttt{PDF, pdf}
Select the \texttt{PDF}-slant engine. This requires the document is translated with pdflatex or a compatible program.

\texttt{posslant=⟨slant-expr⟩}
Set the default value for \texttt{\synthsllant} only. The argument \texttt{⟨slant-expr⟩} is a floating-point expression. Note that for this option \texttt{⟨slant-expr⟩} must evaluate to a non-negative value.

\texttt{PS, ps}
Use PStricks to delegate slanting to the PostScript interpreter. Obviously requires PStricks\(^{11}\) and DVI-to-PostScript translation.

\(^9\) \texttt{Requires fontspec.sty.}
\(^{10}\) \texttt{This option requires l3draw.sty.}
\(^{11}\) \texttt{The package actually required is pst-3d.sty.}
\begin{Caution}
This engine is still experimental and produces low-quality output! See Sec. 5 on p. 15 for details.
\end{Caution}

\texttt{slant=(slant-expr)}

Set the default values for both \texttt{\synthslant} and \texttt{\synthnegslant}, this is, behave as if the two package options \texttt{posslant = (slant-expr)} and \texttt{negslant = -(slant-expr)} have been given. The argument \texttt{(slant-expr)} is a floating-point expression.

If omitted \texttt{\synthslant} assumes \texttt{.2} and \texttt{-.2}, respectively.

\texttt{TikZ, tikz}

Use TikZ for slanting.\footnote{Requires \texttt{tikz.sty}.}

\begin{Caution}
This engine is still experimental and produces low-quality output! See Sec. 5 on p. 15 for details.
\end{Caution}

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

\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.

<table>
<thead>
<tr>
<th>Font Name</th>
<th>Slant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF Accanthis</td>
<td>.26</td>
</tr>
<tr>
<td>ADF Baskerville</td>
<td>.32</td>
</tr>
<tr>
<td>ADF Berenis</td>
<td>.2</td>
</tr>
<tr>
<td>ADF Venturis</td>
<td>.2</td>
</tr>
<tr>
<td>Alegreya</td>
<td>.2</td>
</tr>
<tr>
<td>Arvo</td>
<td>.2</td>
</tr>
<tr>
<td>BaskervilleF</td>
<td>.2</td>
</tr>
<tr>
<td>Bera Serif</td>
<td>.2</td>
</tr>
<tr>
<td>Bitter</td>
<td>.16</td>
</tr>
<tr>
<td>Caladea</td>
<td>.14</td>
</tr>
<tr>
<td>Clara</td>
<td>.24</td>
</tr>
<tr>
<td>Cochineal</td>
<td>.2</td>
</tr>
<tr>
<td>Coelacanth</td>
<td>.2</td>
</tr>
<tr>
<td>Crimson Pro</td>
<td>.2</td>
</tr>
<tr>
<td>Crimson Text</td>
<td>.2</td>
</tr>
</tbody>
</table>

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

\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.

Forward slant some upright glyphs.

\textsynthslant{(text)}

In horizontal mode switch to an upright shape, slant (text) with the slant value stored in \textsynthslant 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 \textsynthslant.

**Use Cases**

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

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

where we show the font modification in conjunction with the babel macro \texttt{\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}}}

3.3 Macros and Environments: Advanced Interface

\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.

\textbf{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
\textbf{Example}
where we have assumed that amsmath has been loaded to bring \DeclareMathOperator into scope.

3.3 Advanced Interface
\textsynthslantbox Slant ⟨text⟩ with an amount of ⟨slant⟩ that can be positive, negative or zero.

\textsynthslantbox{⟨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.

\textbf{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.

\textbf{slantenvironment}(env.) Wrapper around \textsynthslantbox 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 \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.

\begin{example}
Upright italics often look somewhat tight. I like to add some extra tracking to them. So, I simply define the context synthnegslant:

\begin{verbatim}
\SetTracking[context = synthnegslant]
  {encoding = *, shape = it}
  {10}
\end{verbatim}
\end{example}
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\,\,\text{\upshape{“}}\}.

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 $\alpha$ or $\langle \text{slant} \rangle$.

*Note*

The slant angles of different glyphs in the same font may slightly differ from each other. We look for a representative $\langle \text{slant} \rangle$, 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 $\alpha$ to a $\langle \text{slant} \rangle$ by calculating $\sin \alpha$.
   
   If no computer is available, the following formula might help:
   
   $$\langle \text{slant} \rangle = \sin \alpha \approx \frac{11}{630^\circ} \alpha,$$

   where $\alpha$ 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](image)

**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 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 \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.

TikZ engine. The TikZ engine produces some extra space at the ends of the text.
6 Alternative Solution

Here is an alternative to \texttt{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 \texttt{pdflatex}

In \texttt{pdflatex} fonts can be re-mapped in the document preamble with the primitive \texttt{\pdfmapline}; see the \texttt{pdflatex} Reference Manual \cite[Sec. 6.1]{pdfTeXRef} 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:

\begin{verbatim}
⟨tfm-name⟩ ⟨ps-name⟩ ⟨font-flags⟩
"⟨special⟩" ⟨encoding-file⟩ ⟨font-file⟩
\end{verbatim}

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 \texttt{dvips} \cite[Sec. 6.3]{dvips} 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 \texttt{otf}, \texttt{pfb}, or \texttt{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 \texttt{pdftex.map} on my system:

\begin{verbatim}
LibreCsln-Italic-osf-t1–base LibreCsln-Italic
"_AutoEnc..._ReEncodeFont_"
"\lsln...\enc <LibreCsln-Italic.pfb
\end{verbatim}

which I had to break into three lines to make it fit this page. The \texttt{...} 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:

\begin{verbatim}
⟨slant⟩ SlantFont
\end{verbatim}

so for a shear to the left, for example, \(⟨slant⟩ = −12\), which means the font gets slanted by \(−8°\) the \(⟨special⟩\) part becomes
6.2 ALTERNATIVE SOLUTION: Combining \LaTeX{} and dvipdfmx

"_−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 \textit{White Handgloves}
Less angled version \textit{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 \texttt{=} 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.

\begin{verbatim}
\langle \hfill \hfill \rangle
\end{verbatim}

\NeedsTeXFormat{LaTeX2e}[2005/12/01]
\ProvidesPackage{synthslant}[2024/05/07 v0.1 Synthetically Slant glyphs]
\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 $0.2$ approximately corresponds to a slant-angle of $12^\circ$.

\providecommand*{\synthslant}{0.2}

\synthnegslant

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

\providecommand*{\synthnegslant}{-0.2}

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.

\synthslelant\engine Default to automatic selection of the slant engine.

\def\synthslelant{\texttt{-1}}

Expose default forward and backward slant values as package options.

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

Make slant-engine selection configurable.

\DeclareOptionX{auto}{\def\synthslelant{\texttt{-1}}}
\DeclareOptionX{PDF}{\def\synthslelant{\texttt{0}}}
\DeclareOptionX{pdf}{\def\synthslelant{\texttt{0}}}
\DeclareOptionX{l3draw}{\def\synthslelant{\texttt{1}}}
\DeclareOptionX{ps}{\def\synthslelant{\texttt{2}}}

\end{verbatim}
A.3 Slant Engines

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

\ifnum\synthslant<0
\PackageInfo{synthslant}{auto-selecting slant engine}
\fi
\ifpdf
\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
The various slant engine macros are all subsumed under \texttt{\synthslant@shear@box}. So the higher-level code becomes (almost) engine independent.

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

\begin{Verbatim}
\newcommand*{\synthslant@engine@name}{
  \ifcase\synthslant@engine
    \or PDF
    \or l3draw
    \or PSTricks
    \or TikZ
    \or fontspec
    \else null-implementation\fi}
\end{Verbatim}

\subsection{PDF Slant Engine}

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

\begin{Verbatim}
\ifcase\synthslant@engine
  \PackageInfo{synthslant}{shearing done by PDF}
  \newbox{\synthslant@box}
\fi
\end{Verbatim}

\texttt{\synthslant@pdf@shear@box}
A.3.2 l3draw Slant Engine

Using \LaTeX{}X3 may be like cheating on a very high level as the draw subsystem may delegate to the PDF-engine itself. LOL!

\begin{verbatim}
\PackageInfo{synthslant}{shearing delegated to l3draw}
\RequirePackage{l3draw}
\ExplSyntaxOn
\synthslant@latex@shear@box
\end{verbatim}

Slanting implemented with the experimental l3draw subsystem.

\textbf{Anticipated Change}

As soon as the l3kernel offers an x-shear operation (box_\texttt{xshear}:Nn?) we shall ditch this implementation and switch to the one that is tailored to text instead of the current one for graphics.

\begin{verbatim}
\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@}
\draw_suspend_begin:
\kern#1\l_tmpb_dim
\draw_suspend_end:
\dim_set:Nn \l_tmpa_dim {\l_tmpa_dim - #1\l_tmpb_dim}
\draw_suspend_begin:
\kern#1\l_tmpb_dim
\draw_suspend_end:
\dim_set:Nn \l_tmpa_dim {\l_tmpa_dim + #1\l_tmpb_dim}
\draw_box_use:N \l_tmpa_box
\draw_end:
\ExplSyntaxOff
\end{verbatim}

Now typeset the box.

\begin{verbatim}
\ExplSyntaxOff
\end{verbatim}
A.3.3 PSTricks Slant Engine

Shearing via PSTricks works, but exhibits a weird interface.

\begin{verbatim}
\PackageInfo{synthslant}{shearing deferred to PostScript via PSTricks}
\RequirePackage{pst-3d}% \pstilt
\let\synthslant@shear@box=\synthslant@pstricks@shear@box
\end{verbatim}

Package pstricks offers \texttt{\textbackslash pstilt} and \texttt{\textbackslash psTilt} both with typographically sub-optimal outcomes.

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.

\begin{verbatim}
\PackageInfo{synthslant}{shearing by TikZ}
\RequirePackage{tikz}
\let\synthslant@shear@box=\synthslant@tikz@shear@box
\end{verbatim}

A.3.5 fontspec

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

\begin{verbatim}
\PackageInfo{synthslant}{use fontspec's artificial font transformations}
\RequirePackage{fontspec}
\ExplSyntaxOn
\end{verbatim}
A.4 PACKAGE CODE: Generic Slant Code

The L\TeX3 and TikZ engines break lines at unexpected points. Here is a duct-tape solution for them that concretes together the adjacent parts.

\synthslant@nolinebreak

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

\synthslantbox@soft@hyphen

Allow for line breaks at hyphenation opportunities (\textendash\textemdash). 

\def\synthslantbox@soft@hyphen#1\textendash#2\relax{% 
  \synthslant@shear@box{\synthslant@slant@value}{#1} 
  \ifx\relax#2% 
  \relax 
  \fi 
}
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\relax
  \edef\synthslant@slant@value{#1}% definition for compatibility of both branches
\synthslantbox@fontspect@shear@box{\synthslant@slant@value}{#2}\
\else
\fi

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 L\LaTeX{} wizards may have other ideas.

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

\begin{verbatim}
newcommand*{\synthslantbox@right@slant@correction}{% 
dimen0=\fontdimen5\font 
kern\synthslant\dimen0elax
\fi
\}
\end{verbatim}

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

\begin{verbatim}
newcommand*{\slantcontext}{tracking=synthslant}
\end{verbatim}

\textit{\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.

\begin{verbatim}
NewDocumentEnvironment{slantenvironment}{
  {\upshape
    \ifcsdef{microtypecontext}
      {\expandafter\microtypecontext\expandafter{\slantcontext}}
    {}}

  {\ifcsdef{endmicrotypecontext}
    {\endmicrotypecontext}
  {}}

  \synthslantbox@right@slant@correction}
\end{verbatim}

\textit{\textsynthslant} User-level macro to slant some text.

\begin{verbatim}
NewDocumentCommand{\textsynthslant}{m}{
  {\ifmmode
    \synthslantbox{\synthslant}{#1}%
  \else
    \slantenvironment\synthslantbox{\synthslant}{#1}%
    \endslantenvironment%
  \fi}
\end{verbatim}

\textit{\right@negslant@correction} We could play the same trick here as in \textit{\synthslantbox@right@slant@correction} and use \textit{\synthnegslant} instead of \textit{\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
References


[3] BEZOS, JAVIER. Package babel. 2021, \url{https://ctan.org/pkg/babel}. The original author of babel was J. L. BRAAMS.


[6] \LaTeX X3 PROJECT TEAM, ed. \texttt{\LaTeX~Xe font selection}. 2023, \url{https://www.latex-project.org/help/documentation/fntguide.pdf}.

[7] \LaTeX X3 PROJECT, The \texttt{\LaTeX X3 Interfaces}. 2024, \url{https://texdoc.org/serve/interface3/0}.


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.

A
auto (option), 6

B
breakpoint at space, 9

D
disable (option), 6
discretionary hyphen, 9

evironments:
  negslantenvironment, \textit{11, 260}
  slantenvironment, \textit{10, 237}

F

tfont
  ADF Accanthis, 8
  ADF Baskervald, 8
  ADF Berenis, 8
  ADF Venturis, 8
  Alegreya, 8
  Arvo, 3, 8
  axis
    ital, 2
    opsz, 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
  Eriwhon, 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
  MLModern, 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
  \TeX Gyre Pagella, 8
  TX Fonts Serif, 8
  URW Antiqua, 4
  URW Antiqua, 8
INDEX

URW Grotesk, 4
URW Nimbus Roman, 8
Utopia, 8
fontspec (option), 6
\fpeval, 15, 16, 17, 18, 134

L
l3draw (option), 6

N
negslant (option), 6
\negslantcontext, II, 258, 263
\negslantenvironment, 274
negslantenvironment (env.), II, 260

P
package
TikZ, 7
babel, 9
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
possllant, 6
slant, 7
tikz, 7
\PackageError, 35, 39
pdf (option), 6
possllant (option), 6
PostScript, 6
ps, 6

S
shear
angle, 4
transformation, 3
slant
correction, 9
operation, 3
slant (option), 7
\slantcontext, II, 235, 240
\slantenvironment, 251
\slantenvironment (env.), 10, 237
\string, 35, 39, 44, 45
\synthnegslant, 8, 10, 16,
17, 38, 39, 45, 272, 275
\synthsllant, 8, 9, 15, 18, 34, 35,
44, 232, 249, 252
\synthslantbox, 10, 218, 249,
252, 272, 275

T
\textsynthslant, 9, 247
\textsynthuprightitalic, 10, 270
tikz (option), 7

X
\xdef, 15, 16, 17, 18