Stretchy

v0.1

S. Lurp slurper04@gmail.com

1 Introduction

The Stretchy package (stylized as Stretchy) is a package for creating "stretched" symbols. These are symbols which can be arbitrarily stretched in some way, for example the e and t in the Stretchy logo itself. Stretchy is a plain-TEX package, but works with LATEX as well. Stretchy works with both pdfTEX and LuaTEX.

Stretchy only currently supports the Computer Modern Roman 10pt font. Further support may or may not be added in the future.

Stretchy works by injecting PDF code directly into the underlying file in order to draw shapes that connect parts of preexisting glyphs. For example, take the Stretchy logo:

Stretchy

the e is made from two horizontal halves of the e glyph, connected by two bars drawn using PDF code, via `\pdfliteral`s. The PDF code is injected so that the added graphical elements line up at roughly the correct placement, but this is done through measurements of the glyph, not by studying its source. This generally gives good results, but due to the fact that Stretchy mixes PDF path and glyph painting, PDF consumers do not always produce visually pleasing results at all levels of magnification. This is unavoidable (when using the method used by Stretchy) unfortunately.

2 Stretchy Symbols

Stretchy provides the stretchy symbols listed below.

2.1 Repeated symbols

Stretchy provides methods of repeating symbols, while also adding stretched material to it. These symbols are:

`\strtyint {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\strtyintlimits {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`: This prints *N* integral signs, with *sup* and *sub* as superscript and subscript material, respectively. `\strtyint` differs from `\strtyintlimits` regarding where the limits are placed. The former places them next to the symbols, the latter above and below.

For example, `\strtyint{5}{}{{\bb R}^5}` and `\strtyintlimits{5}{}{{\bb R}^5}` prints

$$\int\!\!\!\!\!\int_{\mathbb{R}^5} \quad \int\!\!\!\!\!\int\limits_{\mathbb{R}^5}$$

`\strtyoint {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\strtyointlimits {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`: This prints *N* integral signs with a circle stretched out painted on them. *sup* and *sub* are the superscript and subscript material respectively. `\strtyoint` differs from `\strtyointlimits` regarding where the limits are placed. The former places them next to the symbols, the latter above and below.

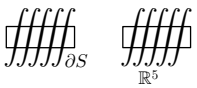
For example, `\strtyoint{5}{}{\partial S}` and `\stryointlimits{5}{}{\partial S}` prints

$$\oint\!\!\!\!\!\int_{\partial S} \quad \oint\!\!\!\!\!\int\limits_{\mathbb{R}^5}$$

`\strtysqint {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\strtysqintlimits {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`: This prints *N* integral signs with a square stretched out painted on them. *sup* and *sub* are the superscript and subscript material respectively. `\strtysqint` differs from

`\strtysqintlimits` regarding where the limits are placed. The former places them next to the symbols, the latter above and below.

For example, `\strtysqint{5}{}{\partial S}` and `\strysqintlimits{5}{}{\partial S}` prints

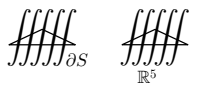


`\strtyrsqint {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\strtyrsqintlimits {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`: This prints N integral signs with a rounded square stretched out painted on them. *sup* and *sub* are the superscript and subscript material respectively. `\strtyrsqint` differs from `\strtyrsqintlimits` regarding where the limits are placed. The former places them next to the symbols, the latter above and below.

For example, `\strtyrsqint{5}{}{\partial S}` and `\stryrsqintlimits{5}{}{\partial S}` prints

`\strtytriint {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\strtytriintlimits {`⟨*N*⟩`}{`⟨*sup*⟩`}{`⟨*sub*⟩`}`: This prints N integral signs with a triangle stretched out painted on them. *sup* and *sub* are the superscript and subscript material respectively. `\strtytriint` differs from `\strtytriintlimits` regarding where the limits are placed. The former places them next to the symbols, the latter above and below.

For example, `\strtytriint{5}{}{\partial S}` and `\strytriintlimits{5}{}{\partial S}` prints



`\pii {`⟨*N*⟩`}`: Prints π with N legs. This can be used for fractions of π. That is, `\pii{N}` corresponds to the value $2\pi/N$. For example, `\pii{5}` gives

2.2 Stretched symbols

Stretchy provides methods of stretching symbols, allowing them to grow arbitrarily large. These symbols are:

`\xint {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`: This draws an integral sign stretched to match the height and depth of *material* with superscript and subscript material corresponding to *sup* and *sub* respectively.

For example `\xint {-3}{-2}{\sum_{n=1}^\infty n^x\,dx}` produces

$$\int_{-2}^{-3} \sum_{n=1}^{\infty} n^x \, dx$$

`\xhsum {`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\xvsum {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`
`\xhvsum {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`:

- `\xhsum` paints a summation symbol stretched horizontally to match the width of its limits;
- `\xvsum` paints a summation symbol stretched vertically to match the height and depth of *material* with the specified limits;
- `\xhvsum` paints a summation symbol stretched both horizontally (to match the width of its limits) and vertically (to match the height and depth of *material*).

For example,

```
\xhsum {}{n\in\{2a+3b\;\mid\;a,b\in{\bb Z}\}}{1\over n}
\xvsum {}{n\in\{2a+3b\;\mid\;a,b\in{\bb Z}\}}{{1\over n}}
\xhvsum {}{n\in\{2a+3b\;\mid\;a,b\in{\bb Z}\}}{{1\over n}}
```

$$\xhbigcup_{n\in\{2a+3b\mid a,b\in\mathbb{Z}\}}\frac{1}{n};\qquad \xvbigcup_{n\in\{2a+3b\mid a,b\in\mathbb{Z}\}}\frac{1}{n};\qquad \xhvbigcup_{n\in\{2a+3b\mid a,b\in\mathbb{Z}\}}\frac{1}{n}$$

`\xhbigcup {`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\xvbigcup {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`
`\xhvbigcup {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`:

- `\xhbigcup` paints a big-cup symbol stretched horizontally to match the width of its limits;
- `\xvbigcup` paints a big-cup symbol stretched vertically to match the height and depth of *material* with the specified limits;
- `\xhvbigcup` paints a big-cup symbol stretched both horizontally (to match the width of its limits) and vertically (to match the height and depth of *material*).

For example,

```
\xhbigcup{}{f\in L^2(\mu),\int f>0}\left\{{1\over f},f\right\};\qquad
\xvbigcup{}{f\in L^2(\mu),\int f>0}{\left\{{1\over f},f\right\}};\qquad
\xhvbigcup{}{f\in L^2(\mu),\int f>0}{\left\{{1\over f},f\right\}}
```

$$\xhbigcup_{f\in L^2(\mu),\int f>0}\left\{\frac{1}{f},f\right\};\qquad \xvbigcup_{f\in L^2(\mu),\int f>0}\left\{\frac{1}{f},f\right\};\qquad \xhvbigcup_{f\in L^2(\mu),\int f>0}\left\{\frac{1}{f},f\right\}$$

`\xhbigcap {`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\xvbigcap {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`
`\xhvbigcap {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`:

- `\xhbigcap` paints a big-cap symbol stretched horizontally to match the width of its limits;
- `\xvbigcap` paints a big-cap symbol stretched vertically to match the height and depth of *material* with the specified limits;
- `\xhvbigcap` paints a big-cap symbol stretched both horizontally (to match the width of its limits) and vertically (to match the height and depth of *material*).

For example,

```
\xhbigcap{}{n=1,2,3,\dots}\left[0,{1\over n}\right]
\xvbigcap{}{n=1,2,3,\dots}{\left[0,{1\over n}\right]}
\xhvbigcap{}{n=1,2,3,\dots}{\left[0,{1\over n}\right]}
```

$$\xhbigcap_{n=1,2,3,\dots}\left[0,\frac{1}{n}\right];\qquad \xvbigcap_{n=1,2,3,\dots}\left[0,\frac{1}{n}\right];\qquad \xhvbigcap_{n=1,2,3,\dots}\left[0,\frac{1}{n}\right]$$

`\xhprod {`⟨*sup*⟩`}{`⟨*sub*⟩`}`
`\xvprod {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`
`\xhvprod {`⟨*sup*⟩`}{`⟨*sub*⟩`}{`⟨*material*⟩`}`:

- `\xhprod` paints a product symbol stretched horizontally to match the width of its limits;
- `\xvprod` paints a productsymbol stretched vertically to match the height and depth of *material* with the specified limits;
- `\xhvprod` paints a product symbol stretched both horizontally (to match the width of its limits) and vertically (to match the height and depth of *material*).

For example,

```
\xhprod{}{n=1,2,3,\dots}\left[0,{1\over n}\right]
\xvprod{}{n=1,2,3,\dots}{\left[0,{1\over n}\right]}
\xhvprod{}{n=1,2,3,\dots}{\left[0,{1\over n}\right]}
```

$$\prod_{n=1,2,3,\dots}\left[0,\frac{1}{n}\right];\qquad \prod_{n=1,2,3,\dots}\left[0,\frac{1}{n}\right];\qquad \prod_{n=1,2,3,\dots}\left[0,\frac{1}{n}\right]$$

2.3 Stretchy Logo

Stretchy also provides macros for producing its logo. To produce the logo itself, Stretchy provides the macro `\stretchylogo`:

Stretchy

To produce the e and t in the Stretchy logo, Stretchy provides the macros `\strty@e` and `\strty@t`, whose usages are

`\strty@e` {⟨*width*⟩}
`\strty@t` {⟨*width*⟩}{⟨*height+depth*⟩}

For example,

```
\strty@e{15pt}
\strty@t{15pt}{10pt}
```

3 Stretchy Internals

Stretchy provides various macros for creating your own stretchy symbols. These all require knowledge regarding the usage of PDF path painting operators. For an in-depth explanation on PDFs and the usage of pdfTEX primitives, you may consult my article here.

3.1 Stretchy Coordinates

All Stretchy painting commands should utilize the Stretchy coordinate system. This is accessed and manipulated via the following macros:

- `\strty@p`;
- `\strty@pd`;
- `\strty@trans`;
- `\strty@setpttrans`.

Essentially, all these macros do is apply a transformation to the coordinates provided. That is, if you specify a line from (x_0, y_0) to (x_1, y_1), Stretchy will transform these coordinates according to the Current Stretchy Transformation (CST) T_S, and draw a line from $T_S(x_0, y_0)$ to $T_S(x_1, y_1)$. The idea is similar for cubic Bézier curves (the start, end, and control points are transformed via T).

In order to facilitate this, you must pass the coordinates to `\strty@p`. That is, instead of doing something like

```
0 0 m
10 0 l
S
```

You should do

```
\strty@p{0}{0} m
\strty@p{10}{0} l
S
```

The CST is specified by `\strty@trans`, which is a macro accepting 2 parameters and must expand to two groups. For example,

```
\def\strty@trans#1#2{{-#2}{#1}}
```

will rotate all points by 90 degrees.
Stretchy provides some useful macros for basic arithmetic operations.

- `\strty@nopt` computes a dimension expression, and expands to the result without the trailing `pt`. For example `\strty@nopt{1pt+2pt}` will expand to `3`.
- `\strty@add` accepts two parameters (numbers), and expands to their sum.
- `\strty@mult` accepts two parameters (numbers), and expands to their product.

The definitions of `\strty@add` and `\strty@mult` are simply

strty-utils.tex

```
\bgroup\lccode`?=`p\lccode`!=`t
\lowercase{\egroup\def\strty@rmpt#1?!{#1}}
```

The definition of `\strty@setpttrans` is simply

strty-utils.tex

```
}}

\def\strty@pttrans{0.996264 0 0 0.996264 0 0 cm}
```

that is, it simply multiplies each component by `\strty@ptm`, which is defined to be `.996264` (the ratio between TEX and PDF pts). Let us define this transformation to be T_{pt}.
`\strty@pd` is a macro accepting four parameters:

`\strty@pd {`⟨*x*⟩`}{`⟨*y*⟩`}{`⟨*dx*⟩`}{`⟨*dy*⟩`}`

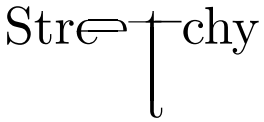
it transforms the point (x, y) to $T_S(x, y) + (dx, dy)$ where dx, dy are dimensions. This is useful e.g. in `\sqrtyrsqint`, where the rounded edges are a set dimension, and thus the vertical edges must be offset by a set dimension.

3.2 Stretchy Utilities

In `strty-utils.tex`, Stretchy defines some useful utilities. Most of these are internal to Stretchy or were discussed previously, but we take the time to discuss one: `\strty@scalebox`. This accepts two parameters:

`\strty@scalebox {`⟨*scale*⟩`}{`⟨*material*⟩`}`

and scales *material* by *scale*. For example `\strty@scalebox{2}{\stretchylogo}` will produce



3.3 Repeated Symbols

In `strty-repeatedsyms.tex`, Stretchy defines all the repeated symbols (listed above) as well as some useful auxillary macros.

`\strty@circle {x}{y}{r}`: This expands to PDF code for drawing a circle centered at (x, y) (dimensions, not affected by the CST) with a radius of r (not a dimension). The axis points of the circle end up being

- right: $T_S(r, 0) + (x, y)$;
- left: $T_S(-r, 0) + (x, y)$;
- top: $T_S(0, r) + (x, y)$;
- bottom: $T_S(0, -r) + (x, y)$.

In order to draw the circle, Stretchy draws four cubic Bézier curves. The value `\strty@cd` determines the distance of the control points from the axis points of the circle.

`\strty@repeatedsum {`⟨*name*⟩`}{`⟨*symbol*⟩`}{`⟨*kerning*⟩`}`: This defines a macro of name *name* which accepts a single parameter N, and paints the symbol *symbol* N times with *kerning* placed between subsequent symbols. For example,

strty-repeatedsyms.tex

```
\strty@repeatedsym{strty@dint@sym}{\displaystyle\int}{\mkern-10mu}
\strty@repeatedsym{strty@tint@sym}{\textstyle\int}{\mkern-7mu}
```

defines the two variants of `\strtyint` and `\strtyintlimits` (one for display math and the other for textstyle math). This defines the macros `\strty@dint@sym` and `\strty@tint@sym`. For example, `\strty@dint@sym{5}` paints

$$\iiiii$$

`\strty@extensible {`⟨*name*⟩`}{`⟨*bg code*⟩`}{`⟨*fg code*⟩`}{`⟨*symbol*⟩`}{`⟨*dx*⟩`}{`⟨*dy*⟩`}{`⟨*kerning*⟩`}`: This defines a macro of name *name* which accepts a single parameter N and paints the symbol *symbol* N times with *kerning* placed between subsequent symbols. *bg code* is PDF code placed before painting the symbols, and *fg code* is PDF code placed after painting the symbols.

If the resulting width of painting N *symbol*s is w, then we define T_S to be

$$T_S\colon (x, y) \mapsto \left(\frac{x(w + dx)}{2}\,,\, y \cdot dy\right)$$

that is, T_S stretches the x-axis by a factor of $\frac{w+dx}{2}$ and the y-axis by a factor of dy.

For example, `\strty@dsqint@sym` is the symbol for `\strtysqint` in display math, and is defined like so:

strty-repeatedsyms.tex

```
\strty@extensible{strty@dsqint@sym}{}{
    q
    .5 w
    1 j 1 J
    \strty@p{-1}{1} m
    \strty@p{1}{1} l
    \strty@p{1}{-1} l
    \strty@p{-1}{-1} l
    s
    Q
}{\displaystyle\int}{-2pt}{4pt}{\mkern-10mu}
```

`\strty@createoplims \`⟨*macro*⟩`{nolim sup kern}{nolim sub kern}{lim sup kern}{lim sub kern}`: This defines two macros, `\macro@nolim` and `\macro@lim` which accept three arguments each, N, *sup* and *sub*. These then pass N to `\macro` (which is a repeating macro, e.g. defined by `\strty@extensible`), and place *sup* and *sub* in the super- and subscripts, with kerning according to the parameters given.

For example, the definition of `\strtyint` and `\strtyintlimits` is

strty-repeatedsyms.tex

```
\strty@repeatedsym{strty@dint@sym}{\displaystyle\int}{\mkern-10mu}
\strty@repeatedsym{strty@tint@sym}{\textstyle\int}{\mkern-7mu}
\strty@createoplims\strty@dint@sym{0mu}{-12mu}{12mu}{-15mu}
\strty@createoplims\strty@tint@sym{0mu}{-7mu}{7mu}{-7mu}

\def\strtyint#1#2#3{%
    \mathchoice%
        {\strty@dint@sym@nolim{#1}{#2}{#3}}%
        {\strty@tint@sym@nolim{#1}{#2}{#3}}%
        {\strty@tint@sym@nolim{#1}{#2}{#3}}%
        {\strty@tint@sym@nolim{#1}{#2}{#3}}%
}

\def\strtyintlimits#1#2#3{%
    \mathchoice%
```

```
76          {\strty@dint@sym@lim{#1}{#2}{#3}}%
77          {\strty@tint@sym@lim{#1}{#2}{#3}}%
78          {\strty@tint@sym@lim{#1}{#2}{#3}}%
79          {\strty@tint@sym@lim{#1}{#2}{#3}}%
80  }
```

(1) lines 61 and 62 define the symbols;

(2) lines 63 and 64 define the kerning of the super and subscripts;

(3) the rest define the actual macros.

3.4 Stretched Symbols

Stretchy provides the following auxillary macros for creating stretched symbols.

`\strty@hstretch {⟨name⟩}{⟨left⟩}{⟨right⟩}{⟨code⟩}`: This defines a macro named *name* which accepts a parameter w, and creates a symbol of width w. This symbol consists of (from left to right):

(1) the material *left*;

(2) the code *code*;

(3) the material *right*.

The CST is set to be the composition of T_{pt} with

$$(x, y) \mapsto ((w - w_l - w_r) \cdot x + w_l\,,\ y)$$

where w_l, w_r are the widths of *left* and *right*, respectively. That is, $(0, 0)$ is mapped to $(w_l, 0)$, $(1, 0)$ is mapped to $(w - w_r, 0)$, and $(0, 1)$ is mapped to $(0, 1)$. So the left side of *code* maps to w_l (the edge of *left*), and the right side to $w - w_r$ (the edge of *right*).

`\strty@vstretch {⟨name⟩}{⟨top⟩}{⟨bottom⟩}{⟨code⟩}`: This defines a macro named *name* which accepts a parameter h, and creates a symbol of height h. This symbol consists of (from top to bottom):

(1) the material *top*;

(2) the code *code*;

(3) the material *bottom*.

The CST is set to be the composition of T_{pt} with

$$(x, y) \mapsto (x\,,\ (d - h_t - h_b) \cdot y + h_b)$$

where h_t, h_b are the heights of *top* and *bottom*, respectively. That is, $(0, 0)$ is mapped to $(0, h_b)$, $(1, 0)$ is mapped to $(1, 0)$, and $(0, 1)$ is mapped to $(0, d - h_t)$. So the top of *code* maps to $d - h_t$ (the bottom of *top*), and the bottom to h_b (the top of *bottom*).

`\strty@hvstretch {⟨name⟩}{⟨tl⟩}{⟨tr⟩}{⟨bl⟩}{⟨br⟩}{⟨top code⟩}{⟨bot code⟩}{⟨mid code⟩}`: This defines a macro named *name* which accepts two parameters w, h. It creates a symbol of width w and height h of the form:

$$\begin{pmatrix} tl & top\ code & tr \\ & mid\ code & \\ bl & bot\ code & br \end{pmatrix}$$

The transformations for each code are as follows:

- *top code*: let w_{tl}, w_{tr} be the widths of *tl* and *tr* respectively. Then T_S is the composition of T_{pt} with

  $$(x, y) \mapsto ((w - w_{tl} - w_{tr}) \cdot x + w_{tl}\,,\ y)$$

  that is, the left side of *top code* maps to w_{tl}, and the right side to $w - w_{tr}$.

- *bot code*: let w_{bl}, w_{br} be the widths of *bl* and *br* respectively. Then T_S is the composition of T_{pt} with

$$(x, y) \mapsto ((w - w_{bl} - w_{br}) \cdot x + w_{bl}\ ,\ y)$$

  that is, the left side of *top code* maps to w_{bl}, and the right side to $w - w_{br}$.

- *mid code*: let h_t, h_b be the heights of the top and bottom materials, respectively. Then T_S is the composition of T_{pt} with

$$(x, y) \mapsto (w \cdot x\ ,\ (h - h_t - h_b) \cdot y + h_b)$$

  so

  - $(0, 0)$ (the bottom left of *mid code*) maps to $(0, h_b)$ (the top left of the bottom material);
  - $(1, 0)$ (the bottom right) maps to (w, h_b) (the top right of the bottom material);
  - $(0, 1)$ (the top left) maps to $(0, h - h_t)$ (the bottom left of the top material);
  - $(1, 1)$ (the top right) maps to $(w, h - h_t)$ (the bottom right of the top material).

To create a stretched symbol, there are four steps:

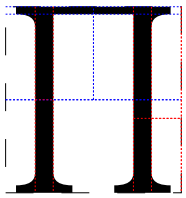
(1) precisely measure the dimensions of the glyph you're making stretchy;

(2) crop the glyph where you want (either using form XObjects, or clipping paths);

(3) use `\strty@XXstretch` (`XX` $\in \{$`h`, `v`, `hv`$\}$) on the cropped sections of the glyph, as well as the code to connect them. This creates a stretchable symbol;

(4) define macro to get the material (super-, sub-script, and main material) and compute the dimensions to stretch the symbol to.

For example, we can measure (textstyle) $\prod$ to get the (rough) dimensions of the strokes



Name	Start	End	Difference
(left stroke)	1.59pt	2.56pt	0.97pt
(right stroke left)	6.87047pt	9.44447pt	2.574pt
(right stroke right)	7.84047pt	9.44447pt	1.604pt

Name	Start	End	Difference
(top stroke top)	5.00006pt	10.0pt	4.99994pt
(top stroke bot)	5.00006pt	9.6pt	4.59995pt

Now we can create our own stretched `\prod`, with the following code:

```
\bgroup
\setbox0=\hbox{\mbnodp{$\prod$}}    % product with no depth
\setbox1=\vbox to.5\ht0{\copy0\vss}
\setbox2=\vbox to.5\ht0{\vss\copy0}
\setbox3=\hbox to.5\wd0{\copy1\hss} % tl
\setbox4=\hbox to.5\wd0{\hss\copy1} % tr
\setbox5=\hbox to.5\wd0{\copy2\hss} % bl
\setbox6=\hbox to.5\wd0{\hss\copy2} % br
\pdfxform3 \xdef\prodtl{\pdfrefxform\the\pdflastxform}
\pdfxform4 \xdef\prodtr{\pdfrefxform\the\pdflastxform}
\pdfxform5 \xdef\prodbl{\pdfrefxform\the\pdflastxform}
\pdfxform6 \xdef\prodbr{\pdfrefxform\the\pdflastxform}
\egroup

\strty@hvstretch{stretchedprod}{\prodtl}{\prodtr}{\prodbl}{\prodbr}%
{   % top connecting horizontal line
    \strty@p{0}{5} m
    \strty@p{1}{5} l
    \strty@p{1}{4.6} l
    \strty@p{0}{4.6} l
```

```
    h
    f
}%
{}%
{%  connecting vertical lines
    \strty@pd{0}{0}{1.59pt}{0pt} m
    \strty@pd{0}{1}{1.59pt}{0pt} l
    \strty@pd{0}{1}{2.56pt}{0pt} l
    \strty@pd{0}{0}{2.56pt}{0pt} l
    h
    f
    \strty@pd{1}{0}{-2.574pt}{0pt} m
    \strty@pd{1}{1}{-2.574pt}{0pt} l
    \strty@pd{1}{1}{-1.604pt}{0pt} l
    \strty@pd{1}{0}{-1.604pt}{0pt} l
    h
    f
}
```

Doing then `\stretchedprod{20pt}{20pt}`, will give, for example:

The rest of the code to get the dimensions of the limits and main material and pass to `\stretchedprod`is standard.