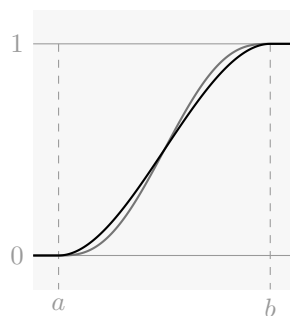


The **easing*** Library for PGF

Loh Ka-tsun[†]



Contents

1	Introduction	2
2	Installation	2
3	Usage	2
3.1	Naming conventions	3
3.2	Specifying parameters	4
4	List of easing function shapes	5
4.1	Polynomials	5
4.1.1	The smooth and smoother shapes	5
4.1.2	The pow shape and friends (linear , quad , cubic , quart , and quint)	5
4.1.3	The back shape	6
4.2	Trigonometric and exponential	6
4.2.1	The sine shape	6
4.2.2	The exp shape	7
4.3	Other	7
4.3.1	The circ shape	7
4.3.2	The elastic shape	8
5	Implementation	9

*version 0.1, dated 2021/07/17. This version was tested with PGF version 3.1.9a

[†]loh.ka.tsun@gmail.com

1 Introduction

This library adds easing functions to the PGF mathematical engine.

2 Installation

The `easing` library is a PGF library; it works both with \LaTeX and with plain \TeX . Once the file `pgflibraryeasing.code.tex` is in a directory searched by \TeX , the library can be loaded as follows:

with plain \TeX :

```
\input pgf
\usepgflibrary{easing}
```

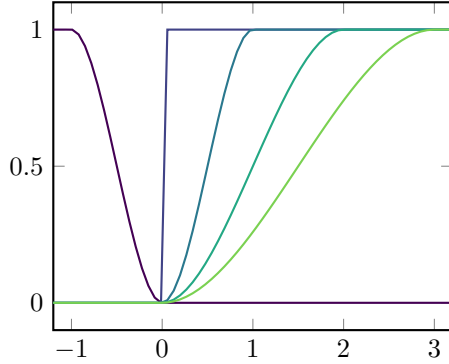
with \LaTeX :

```
\usepackage{pgf}
\usepgflibrary{easing}
```

The `easing` library is compatible with, but does not depend on, the floating point unit library provided by PGF. To use both `easing` and the FPU, the FPU (or any packages/libraries which use the FPU, such as `pgfplots`) must be loaded before the `easing` library.

3 Usage

The routines implemented by the `easing` library are added to PGF's mathematical engine with `\pgfmathdeclarefunction`, so that they are recognised by `\pgfmathparse` and can be used in any expression which is processed by the parser. As a first example, the following code produces plots of the function `smoothstep(a,b,x)` against the argument x , with one endpoint $a = 0$ and the other endpoint b ranging through the integers -1 to 3 :



```

\input pgfplots
\usepgflibrary{easing}
\tikzpicture
\axis[
  domain=-1.2:3.2, samples=64,
  xmin=-1.2, xmax=3.2,
  cycle list={
    [samples of colormap=6 of viridis]},
  no marks, thick]
\pgfplotsinvokeforeach{-1,...,3}{
  \addplot{smoothstep(0,#1,x)};}
\endaxis
\endtikzpicture
\end

```

(This example also demonstrates the behaviour of the easing functions in some special cases: when the endpoints $b \leq a$, and in particular the degenerate case where $a = b$, in which the library chooses to consider the function that is 1 for all $x \geq 0$ and 0 otherwise.)

Like all functions declared in this way, the functions implemented by `easing` are also available as “public” macros, such as `\pgfmathsmoothstep`:

$S_1(0) = 0.0$
 $S_1(0.25) = 0.15625$
 $S_1(0.5) = 0.5$
 $S_1(0.75) = 0.84375$
 $S_1(1) = 1.0$

```

\input pgf
\usepgflibrary{easing}
\foreach\x in{0,0.25,...,1}{
  \pgfmathsmoothstep{0}{1}{\x}
  $S_1(\x)=\pgfmathresult$par
}
\end

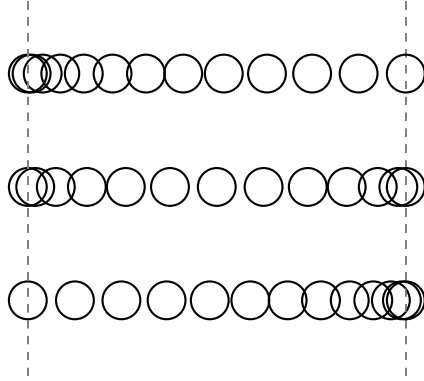
```

See Part VIII of the PGF manual for more details on the mathematical engine.

3.1 Naming conventions

For each shape, three functions are declared, all of which take three arguments a, b , and x . Where $a < b$, all of these function take value 0 whenever $x \leq a$ and 1 whenever $x \geq b$. The names of the functions adhere to the following pattern:

- The *ease-in* form $\langle shape \rangle \text{easein}(a,b,x)$ has easing applied near the endpoint a .
- The *ease-out* form $\langle shape \rangle \text{easeout}(a,b,x)$ has easing applied near the endpoint b . Its graph is that of the ease-in form reflected about both axes.
- The *step function* form $\langle shape \rangle \text{step}(a,b,x)$ has easing applied near both endpoints. Its graph is that of the ease-in and ease-out forms concatenated then appropriately scaled.



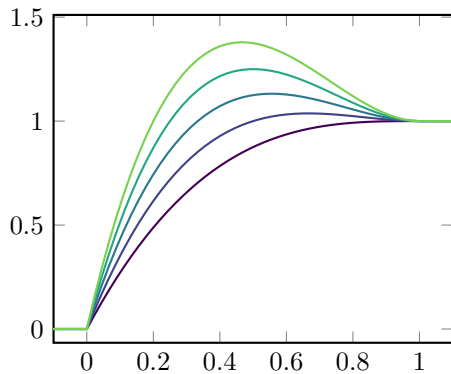
```

\input tikz
\usepgflibrary{easing}
\tikzpicture
\foreach\x in{0,...,12}{
  \draw[gray,dashed]
    (0,-1) -- (0,4) (5,-1) -- (5,4);
  \draw[thick]
    ({5*smootheasein(0,12,\x)},3)
    circle (0.25)
    ({5*smoothstep(0,12,\x)},1.5)
    circle (0.25)
    ({5*smootheaseout(0,12,\x)},0)
    circle (0.25);
}
\endtikzpicture
\end

```

3.2 Specifying parameters

Some of these shapes can be modified by adjusting one or more parameters, which is done through **pgfkeys**: the parameter $\langle param \rangle$ for functions of shape $\langle shape \rangle$ is specified by setting the PGF key `/easing/ $\langle shape \rangle$ / $\langle param \rangle$` :



```

\input pgfplots
\usepgflibrary{easing}
\tikzpicture
\axis[
  domain=-0.2:1.2, samples=64,
  xmin=0, xmax=1, enlarge x limits,
  cycle list={
    [samples of colormap=6 of viridis]},
  no marks, thick]
\pgfplotsinvokeforeach{0,...,4}{
  \pgfkeys{easing,back/overshoot=#1}
  \addplot{backeaseout(0,1,x)};
}
\endaxis
\endtikzpicture
\end

```

Setting a parameter affects the ease-in, step, and ease-out forms of the relevant function at once.

For detailed descriptions of the parameters admitted by each shape, see the following section.

4 List of easing function shapes

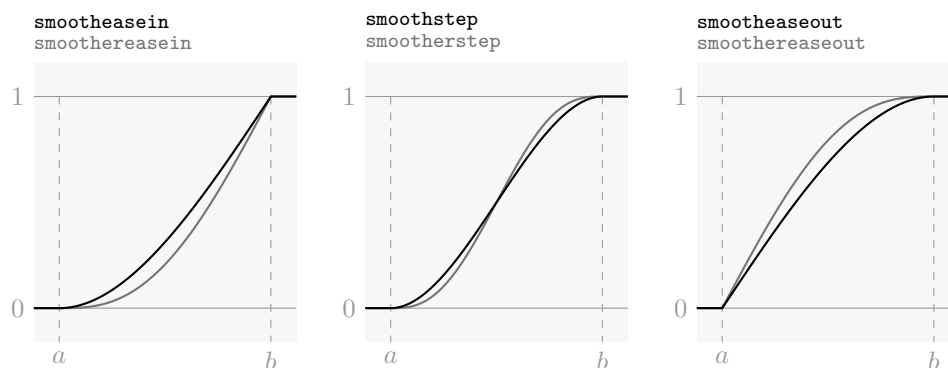
An exhaustive list follows of all the easing functions implemented by the `easing` library. For clarity, where mathematical expressions are given for functions, they are written in terms of a parameter t equal to $\frac{x}{b-a}$.

4.1 Polynomials

4.1.1 The smooth and smoother shapes

The step function form of the `smooth` shape is a third-order Hermite polynomial interpolation between 0 and 1, so that the first derivate at the endpoints are zero. It is defined $3t^2 - 2t^3$ for $0 \leq t \leq 1$.

The step function form of the `smoother` shape is a fifth-order Hermite polynomial interpolation between 0 and 1, so that the first and second derivatives at the endpoints are zero. It is defined $10t^3 - 15t^4 + 6t^5$ for $0 \leq t \leq 1$.

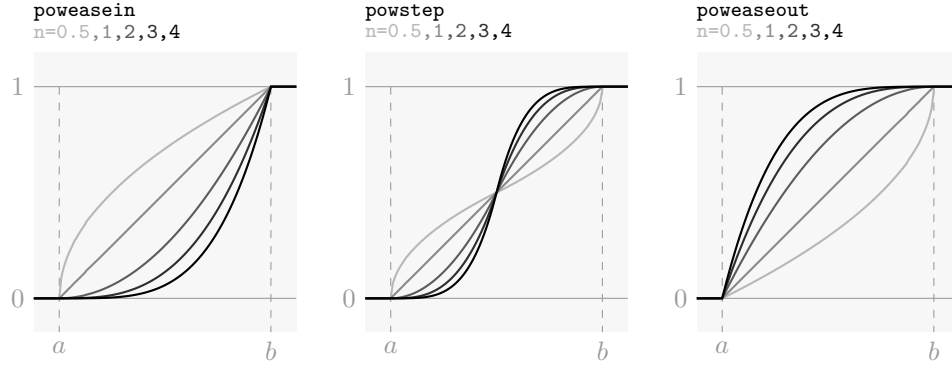


4.1.2 The pow shape and friends (linear, quad, cubic, quart, and quint)

Polynomial easing. The ease-in form is defined as t^n for $0 \leq t \leq 1$, where the exponent n is set with the PGF key `/easing/pow/exponent`, and should be greater than 0. The parameter defaults to $n = 2.4$.

When $n = 1$, the function is linear between 0 and 1. For $0 < n \leq 1$, the ease-in form has discontinuous derivative at 0.

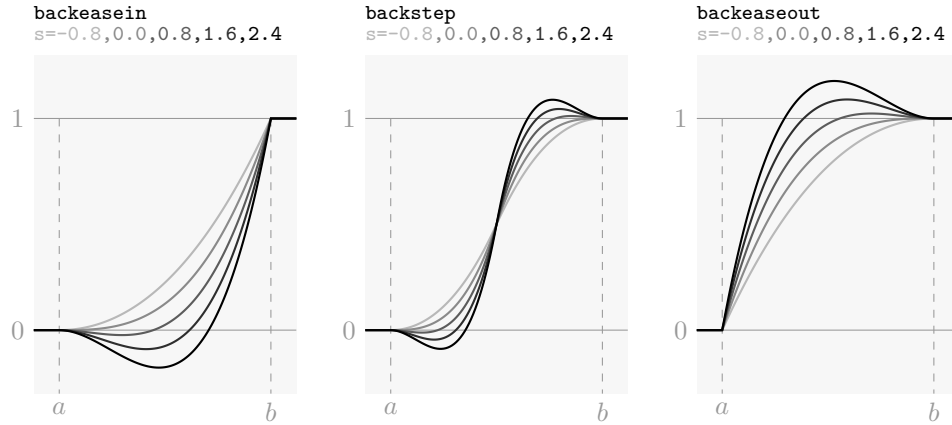
The shapes `linear`, `quad`, `cubic`, `quart`, and `quint` are the same functions as `pow` with $n = 1, \dots, 5$, respectively. Computations for these shapes are implemented with `TEX` registers, which is a little faster and more accurate than setting the argument then evaluating the equivalent `pow` function.



4.1.3 The back shape

Anticipatory easing. The ease-in form is defined as $t^2(1-t)s + t^3$ for $0 \leq t \leq 1$, where the parameter s is set with the PGF key `/easing/back/overshoot`. The parameter defaults to $s = 1.6$.

When $s \leq 0$, there is no overshoot. When $s = 0$, the function is equivalent to `pow` with $n = 3$.

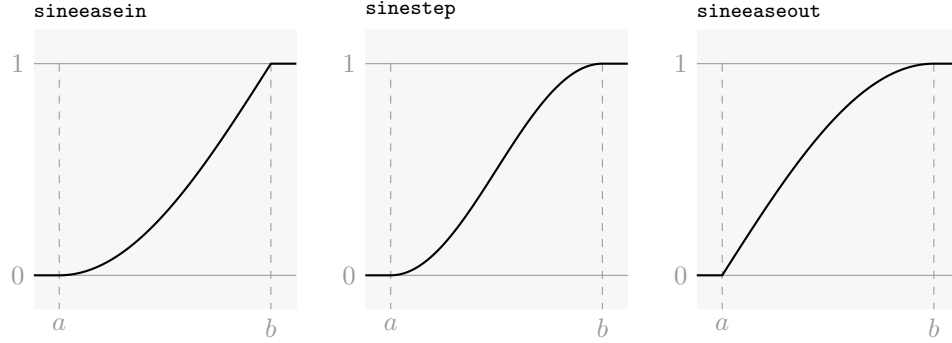


4.2 Trigonometric and exponential

4.2.1 The sine shape

An easing function that looks like a section of a sinusoid. The ease-out form is defined as $\sin(\frac{\pi}{2}t)$ for $0 \leq t \leq 1$.

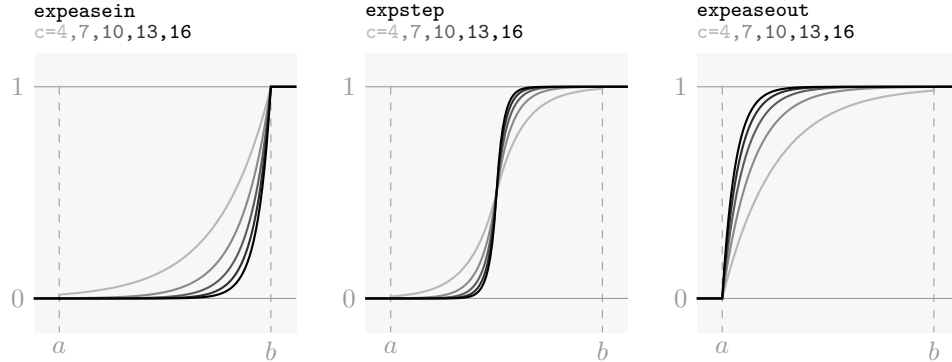
This shape admits no parameters.



4.2.2 The exp shape

An easing function that looks like an exponential function. The ease-in form is defined as $e^{c(t-1)}$ for $0 \leq t \leq 1$, where the parameter c is set with the PGF key `/easing/exp/speed`, and should be greater than 0. The parameter defaults to $c = 7.2$.

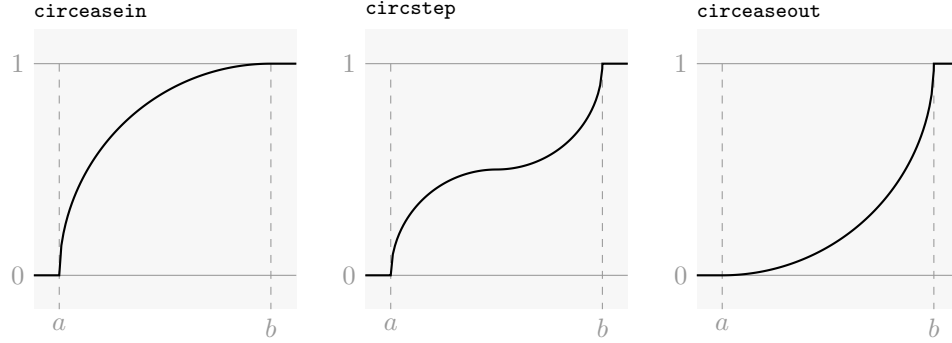
Because of the nature of the exponential function, this shape is only approximately continuous at the endpoints a and b . In practice, the discontinuity only becomes noticeable for small c , around $c \leq 4$.



4.3 Other

4.3.1 The circ shape

An easing function whose graph is part of an ellipse. This shape admits no parameters.

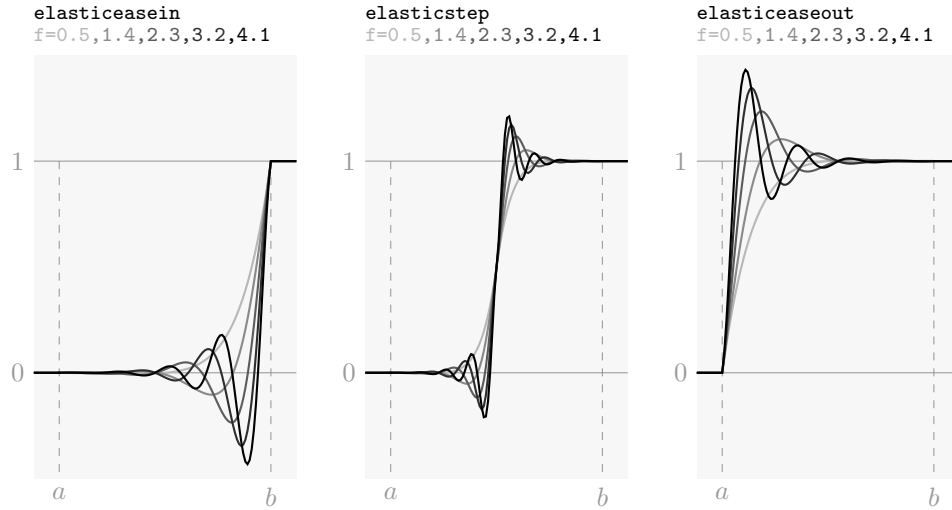


4.3.2 The elastic shape

Easing function that looks like a damped harmonic oscillator. The ease-out form is defined as $e^c(t-1)\cos(2\pi f(1-t))$. This shape admits two parameters:

- The frequency f is the number of oscillations between the endpoints. It is set with the PGF key `/easing/elastic/frequency`, and should be greater than 0. The frequency defaults to $f = 3$.
- The damping coefficient b affects the speed at which the amplitude decays. It is set with the PGF key `/easing/elastic/damping`, and should be greater than zero. The damping coefficient defaults to $b = 7.2$.

The function overshoots the range $[0, 1]$ when $f > 0.5$. For $0 < f \leq 1$, this function becomes a family of anticipatory easing curves that look slightly different from the `back` shape but are more expensive to compute.



5 Implementation

`\ifeasing@withfpu` This library uses \TeX registers and PGF's mathematical engine for computations.
`\easing@divide`

It is possible that the user is loading this library together with the floating point unit library. We save the basic routines from `pgfmath`, so that when this happens, the FPU doesn't break everything when it does a switcharoo with the `pgfmath` macros.

```

1 \newif\ifeasing@withfpu
2 \expandafter\ifx\csname pgflibraryfpuifactive\endcsname\relax
3 \easing@withfpufalse
4 \else
5 \easing@withfptrue
6 \fi
7 \ifeasing@withfpu
8 \let\easing@cos\pgfmath@basic@cos@
9 \let\easing@divide\pgfmath@basic@divide@
10 \let\easing@exp\pgfmath@basic@exp@
11 \let\easing@ln\pgfmath@basic@ln@
12 \let\easing@sqrt\pgfmath@basic@sqrt@
13 \else
14 \let\easing@cos\pgfmathcos@
15 \let\easing@divide\pgfmathdivide@
16 \let\easing@exp\pgfmathexp@
17 \let\easing@ln\pgfmathln@
18 \let\easing@sqrt\pgfmathsqrt@
19 \fi

```

`\easing@linearstep@ne` In absence of the FPU, the next section of code defines `\easing@linearstep`,
`\easing@linearstep@fixed` which expects as arguments plain numbers (i.e. things that can be assigned to
`\easing@linearstep@float` dimension registers). The net effect of `\easing@linearstep{#1}{#2}{#3}` is to
`\easing@linearstep` set `\pgfmathresult` to $\frac{\#3-\#1}{\#2-\#1}$, clamped to between 0 and 1.

If the FPU is loaded, `\easing@linearstep` is instead named `\easing@linearstep@fixed`, and we additionally define `\easing@linearstep@float`, which expects FPU-format floats as arguments. We do not format the output as a float since the FPU is smart enough to do that conversion quietly on its own.

The `\easing@linearstep` routine is the first step in the definition of all other routines that compute easing functions.

```

20 \def\easing@linearstep@ne#1{%
21   \begingroup
22   \pgf@x#1pt
23   \ifdim1pt<\pgf@x\pgf@x 1pt\fi
24   \ifdim0pt>\pgf@x\pgf@x 0pt\fi
25   \pgfmathreturn\pgf@x
26 \endgroup
27 }%

```

```

28 \expandafter\def
29 \csname easing@linearstep\ifeasing@withfpu @fixed\fi\endcsname#1#2#3{%
30   \begingroup
31   \pgf@xa#3pt
32   \pgf@xb#2pt
33   \pgf@xc#1pt
34   \ifdim\pgf@xb=\pgf@xc
35   \edef\pgfmathresult{\ifdim\pgf@xa>\pgf@xb 1\else 0\fi}%
36   \else
37   \advance\pgf@xa-\pgf@xc
38   \advance\pgf@xb-\pgf@xc
39   \easing@divide{\pgfmath@tonumber\pgf@xa}{\pgfmath@tonumber\pgf@xb}%
40   \easing@linearstep@ne\pgfmathresult
41   \fi
42   \pgfmathsmuggle\pgfmathresult
43   \endgroup
44 }%
45 \ifeasing@withfpu
46 \def\easing@linearstep@float#1#2#3{%
47   \begingroup
48   \pgfmathfloatsubtract{#3}{#1}%
49   \edef\pgf@tempa{\pgfmathresult}%
50   \pgfmathfloatsubtract{#2}{#1}%
51   \edef\pgf@tempb{\pgfmathresult}%
52   \pgfmathfloatifflags{\pgf@tempb}{0}{%
53     \pgfmathfloatifflags{\pgf@tempa}{-}{%
54       \edef\pgfmathresult{0}%
55     }{%
56       \edef\pgfmathresult{1}%
57     }%
58   }{%
59     \pgfmathfloatdivide\pgf@tempa\pgf@tempb
60     \pgfmathfloattofixed{\pgfmathresult}%
61     \easing@linearstep@ne\pgfmathresult
62   }%
63   \pgfmathsmuggle\pgfmathresult
64   \endgroup
65 }%
66 \def\easing@linearstep#1#2#3{%
67   \pgflibraryfpuifactive{%
68     \easing@linearstep@float{#1}{#2}{#3}}{%
69     \easing@linearstep@fixed{#1}{#2}{#3}}%
70 }%
71 \fi

```

\easing@linearstep@easein@ne The linear ease-in and ease-out functions are identical to the linear step function.
\easing@linearstep@easeout@ne We define the respective macros so as not to surprise the user with their absence.

```

72 \let\easing@lineareasein\easing@linearstep
73 \pgfmathdeclarefunction{lineareasein}{3}{%

```

```

74 \easing@lineareasein{#1}{#2}{#3}}%
75 \let\easing@lineareaseout\easing@linearstep
76 \pgfmathdeclarefunction{lineareaseout}{3}{%
77 \easing@lineareasein{#1}{#2}{#3}}%

```

```

\easing@derive@easein@nefromstep@ne
\easing@derive@easeout@nefromstep@ne
\easing@derive@step@nefromeasein@ne
\easing@derive@easeout@nefromeasein@ne

```

The pattern in general is that, for each shape, we define the one-parameter version of the step, ease-in, and ease-out routines interpolating between values 0 at 1 at the ends of the unit interval. Then by composing with `\easing@linearstep`, we obtain the three-parameter versions that allow the user to specify the begin and end points of the interpolation.

Most of the time it suffices to define just one of the three one-parameter versions of a shape to be able to infer the form of all three. This is done with the `\easing@derive-from-` macros.

```

78 \def\easing@derive@easein@nefromstep@ne#1{%
79 \expandafter\def\csname easing@#1easein@ne\endcsname##1{%
80 \begingroup
81 \pgf@x##1 pt
82 \divide\pgf@x 2
83 \csname easing@#1step@ne\endcsname{\pgfmath@tonumber\pgf@x}%
84 \pgf@x\pgfmathresult pt
85 \multiply\pgf@x 2
86 \pgfmathreturn\pgf@x
87 \endgroup
88 }%
89 }%
90 \def\easing@derive@easeout@nefromstep@ne#1{%
91 \expandafter\def\csname easing@#1easeout@ne\endcsname##1{%
92 \begingroup
93 \pgf@x##1 pt
94 \divide\pgf@x 2
95 \advance\pgf@x 0.5pt
96 \csname easing@#1step@ne\endcsname{\pgfmath@tonumber\pgf@x}%
97 \pgf@x\pgfmathresult pt
98 \multiply\pgf@x 2
99 \advance\pgf@x -1pt
100 \pgfmathreturn\pgf@x
101 \endgroup
102 }%
103 }%
104 \def\easing@derive@step@nefromeasein@ne#1{%
105 \expandafter\def\csname easing@#1step@ne\endcsname##1{%
106 \begingroup
107 \pgf@x##1 pt
108 \multiply\pgf@x 2
109 \ifdim\pgf@x<1pt
110 \csname easing@#1easein@ne\endcsname{\pgfmath@tonumber\pgf@x}%
111 \pgf@x\pgfmathresult pt
112 \divide\pgf@x 2

```

```

113 \else
114 \multiply\pgf@x -1
115 \advance\pgf@x 2pt
116 \csname easing@#1easein@ne\endcsname{\pgfmath@tonumber\pgf@x}%
117 \pgf@x\pgfmathresult pt
118 \divide\pgf@x 2
119 \multiply\pgf@x -1
120 \advance\pgf@x 1pt
121 \fi
122 \pgfmathreturn\pgf@x
123 \endgroup
124 }%
125 }%
126 \def\easing@derive@easeout@nefromeasein@ne#1{%
127 \expandafter\def\csname easing@#1easeout@ne\endcsname##1{%
128 \begingroup
129 \pgf@x##1pt
130 \multiply\pgf@x -1
131 \advance\pgf@x 1pt
132 \csname easing@#1easein@ne\endcsname{\pgfmath@tonumber\pgf@x}%
133 \pgf@x\pgfmathresult pt
134 \multiply\pgf@x -1
135 \advance\pgf@x 1pt
136 \pgfmathreturn\pgf@x
137 \endgroup
138 }%
139 }%

```

`\easing@pgfmathinstall` The three-parameter versions of each routine is installed into the mathematical engine, so that they are available in `\pgfmathparse`.

```

140 \def\easing@pgfmathinstall#1{%
141 \pgfmathdeclarefunction{#1step}{3}{%
142 \easing@linearstep{##1}{##2}{##3}%
143 \csname easing@#1step@ne\endcsname\pgfmathresult
144 }%
145 \pgfmathdeclarefunction{#1easein}{3}{%
146 \easing@linearstep{##1}{##2}{##3}%
147 \csname easing@#1easein@ne\endcsname\pgfmathresult
148 }%
149 \pgfmathdeclarefunction{#1easeout}{3}{%
150 \easing@linearstep{##1}{##2}{##3}%
151 \csname easing@#1easeout@ne\endcsname\pgfmathresult
152 }%
153 }%

```

`\easing@smoothstep@ne` The smooth shape.
`\easing@smootheasein@ne`
`\easing@smootheaseout@ne`

```

154 \def\easing@smoothstep@ne#1{%
155 \begingroup

```

```

156 \pgf@x#1pt
157 \edef\pgf@temp{\pgfm@th@tonumber\pgf@x}%
158 \multiply\pgf@x-2
159 \advance\pgf@x 3pt
160 \pgf@x\pgf@temp\pgf@x
161 \pgf@x\pgf@temp\pgf@x
162 \pgfm@threturn\pgf@x
163 \endgroup
164 }%
165 \easing@derive@easein@nefromstep@ne{smooth}%
166 \easing@derive@easeout@nefromstep@ne{smooth}%
167 \easing@pgfm@thinstall{smooth}%

```

\easing@smootherstep@ne The smoother shape.

```

\easing@smoothereasein@ne
\easing@smoothereaseout@ne
168 \def\easing@smootherstep@ne#1{%
169 \begingroup
170 \pgf@x#1pt
171 \edef\pgf@temp{\pgfm@th@tonumber\pgf@x}%
172 \multiply\pgf@x 6
173 \advance\pgf@x -15pt
174 \pgf@x\pgf@temp\pgf@x
175 \advance\pgf@x 10pt
176 \pgf@x\pgf@temp\pgf@x
177 \pgf@x\pgf@temp\pgf@x
178 \pgf@x\pgf@temp\pgf@x
179 \pgfm@threturn\pgf@x
180 \endgroup
181 }%
182 \easing@derive@easein@nefromstep@ne{smoother}%
183 \easing@derive@easeout@nefromstep@ne{smoother}%
184 \easing@pgfm@thinstall{smoother}%

```

\easing@powstep@ne The pow shape.

\easing@poweasein@ne
\easing@poweaseout@ne

Because of some wonkiness in the FPU, \pgfm@th@pow@basic@ actually doesn't work. Instead of invoking the pow function, we compute t^n approximately by computing $e^{n \ln t}$ using ln and exp instead (which is what pgfm@th does anyway when the exponent is not an integer.)

```

185 \pgfkeys{/easing/.is family}%
186 \pgfkeys{easing,
187 pow/exponent/.estore in=\easing@param@pow@exponent,
188 pow/exponent/.default=2.4,
189 pow/exponent}%
190 \def\easing@poweasein@ne#1{%
191 \begingroup
192 \pgf@x#1pt
193 \ifdim\pgf@x=0pt
194 \edef\pgfm@thresult{0}%

```

```

195 \else
196 \easing@ln{#1}%
197 \pgf@x\pgfmathresult pt
198 \pgf@x\easing@param@pow@exponent\pgf@x
199 \easing@exp{\pgfmath@tonumber\pgf@x}%
200 \fi
201 \pgfmathsmuggle\pgfmathresult
202 \endgroup
203 }%
204 \easing@derive@easeout@nefromeasein@ne{pow}%
205 \easing@derive@step@nefromeasein@ne{pow}%
206 \easing@pgfmathinstall{pow}%

```

\easing@quadstep@ne The quad-, cubic-, quart-, and quint- routines have explicit definitions.

```

\easing@quadeasein@ne
\easing@quadeaseout@ne
207 \def\easing@quadeasein@ne#1{%
208 \begingroup
209 \pgf@x#1pt
210 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
211 \pgf@x\pgf@temp\pgf@x
212 \pgfmathreturn\pgf@x
213 \endgroup
214 }%
215 \easing@derive@step@nefromeasein@ne{quad}%
216 \easing@derive@easeout@nefromeasein@ne{quad}%
217 \easing@pgfmathinstall{quad}%
218 %
219 \def\easing@cubiceasein@ne#1{%
220 \begingroup
221 \pgf@x#1pt
222 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
223 \pgf@x\pgf@temp\pgf@x
224 \pgf@x\pgf@temp\pgf@x
225 \pgfmathreturn\pgf@x
226 \endgroup
227 }%
228 \easing@derive@step@nefromeasein@ne{cubic}%
229 \easing@derive@easeout@nefromeasein@ne{cubic}%
230 \easing@pgfmathinstall{cubic}%
231 %
232 \def\easing@quarteasein@ne#1{%
233 \begingroup
234 \pgf@x#1pt
235 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
236 \pgf@x\pgf@temp\pgf@x
237 \pgf@x\pgf@temp\pgf@x
238 \pgf@x\pgf@temp\pgf@x
239 \pgfmathreturn\pgf@x
240 \endgroup
241 }%

```

```

242 \easing@derive@step@nefromeasein@ne{quart}%
243 \easing@derive@easeout@nefromeasein@ne{quart}%
244 \easing@pgfmathinstall{quart}%
245 %
246 \def\easing@quinteasein@ne#1{%
247   \begingroup
248   \pgf@x#1pt
249   \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
250   \pgf@x\pgf@temp\pgf@x
251   \pgf@x\pgf@temp\pgf@x
252   \pgf@x\pgf@temp\pgf@x
253   \pgf@x\pgf@temp\pgf@x
254   \pgfmathreturn\pgf@x
255   \endgroup
256 }%
257 \easing@derive@step@nefromeasein@ne{quint}%
258 \easing@derive@easeout@nefromeasein@ne{quint}%
259 \easing@pgfmathinstall{quint}%

```

```

\easing@backstep@ne   The back shape.
\easing@backeasein@ne
\easing@backeaseout@ne
260 \pgfkeys{easing,
261   back/overshoot/.estore in=\easing@param@back@overshoot,
262   back/overshoot/.default=1.6,
263   back/overshoot}%
264 \def\easing@backeasein@ne#1{%
265   \begingroup
266   \pgf@x#1pt
267   \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
268   \advance\pgf@x -1pt
269   \pgf@x\easing@param@back@overshoot\pgf@x
270   \advance\pgf@x\pgf@temp pt
271   \pgf@x\pgf@temp\pgf@x
272   \pgf@x\pgf@temp\pgf@x
273   \pgfmathreturn\pgf@x
274   \endgroup
275 }%
276 \easing@derive@step@nefromeasein@ne{back}%
277 \easing@derive@easeout@nefromeasein@ne{back}%
278 \easing@pgfmathinstall{back}%

```

```

\easing@sinestep@ne   The sine shape.
\easing@sineeasein@ne
\easing@sineeaseout@ne

```

We write down both the `easein` and `step` forms of this, since they are simple compared to what would have been obtained by `\easing@derive-`.

```

279 \def\easing@sineeasein@ne#1{%
280   \begingroup
281   \pgf@x#1pt
282   \multiply\pgf@x 90

```

```

283 \easing@cos{\pgfmath@tonumber\pgf@x}%
284 \pgf@x\pgfmathresult pt
285 \multiply\pgf@x -1
286 \advance\pgf@x 1pt
287 \pgfmathreturn\pgf@x
288 \endgroup
289 }%
290 \def\easing@sinestep@ne#1{%
291 \beginpgfgroup
292 \pgf@x#1pt
293 \multiply\pgf@x 180
294 \easing@cos{\pgfmath@tonumber\pgf@x}%
295 \pgf@x\pgfmathresult pt
296 \divide\pgf@x 2
297 \multiply\pgf@x -1
298 \advance\pgf@x 0.5pt
299 \pgfmathreturn\pgf@x
300 \endgroup
301 }%
302 \easing@derive@easeout@nefromeasein@ne{sine}%
303 \easing@pgfmathinstall{sine}%

```

```

\easing@expstep@ne The exp shape.
\easing@expeasein@ne
\easing@expeaseout@ne 304 \pgfkeys{easing,
305 exp/speed/.estore in=\easing@param@exponent@speed,
306 exp/speed/.default=7.2,
307 exp/speed}%
308 \def\easing@expeasein@ne#1{%
309 \beginpgfgroup
310 \pgf@x#1pt
311 \advance\pgf@x -1pt
312 \pgf@x\easing@param@exponent@speed\pgf@x
313 \easing@exp{\pgfmath@tonumber\pgf@x}%
314 \pgfmathsmuggle\pgfmathresult
315 \endgroup
316 }%
317 \easing@derive@step@nefromeasein@ne{exp}%
318 \easing@derive@easeout@nefromeasein@ne{exp}%
319 \easing@pgfmathinstall{exp}%

```

```

\easing@circstep@ne The circ shape.
\easing@circeasein@ne
\easing@circeaseout@ne 320 \def\easing@circeasein@ne#1{%
321 \beginpgfgroup
322 \pgf@x#1pt
323 \advance\pgf@x -1pt
324 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
325 \pgf@x\pgf@temp\pgf@x
326 \multiply\pgf@x -1

```



```

327 \advance\pgf@x 1pt
328 \easing@sqrt{\pgfmath@tonumber\pgf@x}%
329 \pgfmathsmuggle\pgfmathresult
330 \endgroup
331 }%
332 \easing@derive@step@nefromeasein@ne{circ}%
333 \easing@derive@easeout@nefromeasein@ne{circ}%
334 \easing@pgfmathinstall{circ}%

\easing@elasticstep@ne The elastic shape.
\easing@elasticeasein@ne
\easing@elasticeaseout@ne 335 \pgfkeys{easing,
336 elastic/frequency/.estore in=\easing@param@elastic@frequency,
337 elastic/damping/.estore in=\easing@param@elastic@damping,
338 elastic/frequency/.default=3,
339 elastic/damping/.default=7.2,
340 elastic/frequency, elastic/damping}%
341 \def\easing@elasticeasein@ne#1{%
342 \begingroup
343 \pgf@xa#1pt
344 \advance\pgf@xa -1pt
345 \pgf@xb-\pgf@xa
346 \pgf@xa\easing@param@elastic@damping\pgf@xa
347 \easing@exp{\pgfmath@tonumber\pgf@xa}%
348 \pgf@xa\pgfmathresult pt
349 \pgf@xb 360\pgf@xb
350 \pgf@xb\easing@param@elastic@frequency\pgf@xb
351 \easing@cos{\pgfmath@tonumber\pgf@xb}%
352 \pgf@xa\pgfmathresult\pgf@xa
353 \pgfmathreturn\pgf@xa
354 \endgroup
355 }%
356 \easing@derive@step@nefromeasein@ne{elastic}%
357 \easing@derive@easeout@nefromeasein@ne{elastic}%
358 \easing@pgfmathinstall{elastic}%

```

Change History

0.1
General: Initial version 1