

# The `easing` Library for PGF

Loh Ka-tsun

July 16, 2021

## 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  $\text{\LaTeX}$  and with plain  $\text{\TeX}$ . Once the file `pgflibraryeasing.code.tex` is in a directory searched by  $\text{\TeX}$ , the library can be loaded as follows:

with plain  $\text{\TeX}$

---

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

---

with  $\text{\LaTeX}$ :

---

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

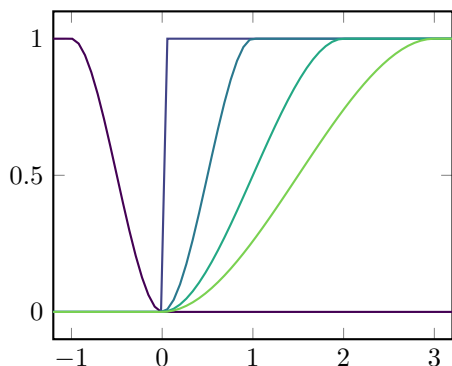
---

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$
}
\end

```

---

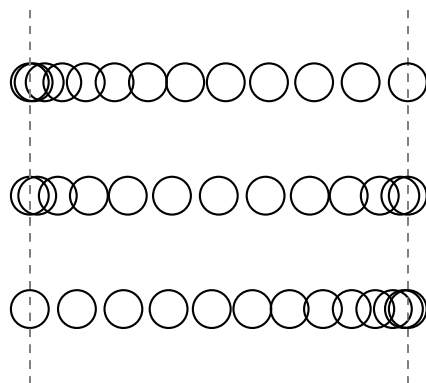
See Part VIII of the PGF manual for more details on the mathematical engine.

### 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 left endpoint  $a$ .

- The *ease-out* form  $\langle shape \rangle \text{easeout}(a,b,x)$  has easing applied near the right 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.




---

```

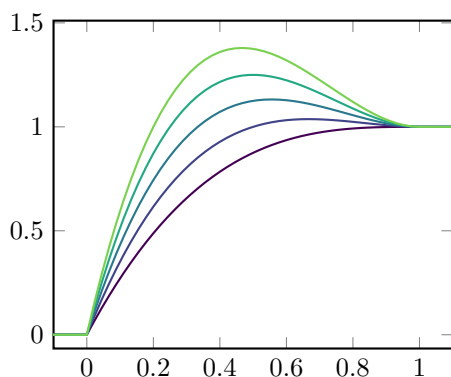
\input tikz
\usepgflibrary{easing}
\tikzpicture
\foreach\x in{0,...,12}{
  \draw[gray,dashed]
    (0,-1) -- (0,4) (5,-1) -- (5,4);
  \draw[thick]
    ({5*smootheasein(0,12,\x)},3)
    circle (0.25)
    ({5*smoothstep(0,12,\x)},1.5)
    circle (0.25)
    ({5*smootheaseout(0,12,\x)},0)
    circle (0.25);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

```

---

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

## 4 List of easing function shapes

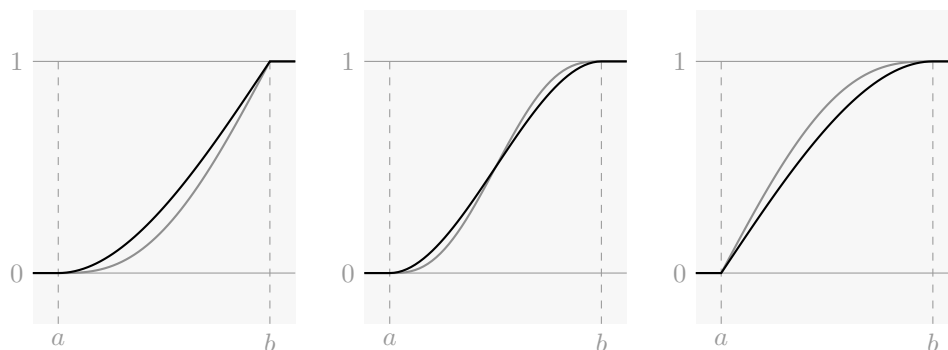
An exhaustive list follows of all the easing functions implemented by the `easing` library. For clarity, where mathematical expressions are given for functions, they are written in terms of a parameter  $t$  equal to  $\frac{x}{b-a}$ .

### 4.1 Polynomial and trigonometric

#### 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  $0 \leq t \leq 1$ .

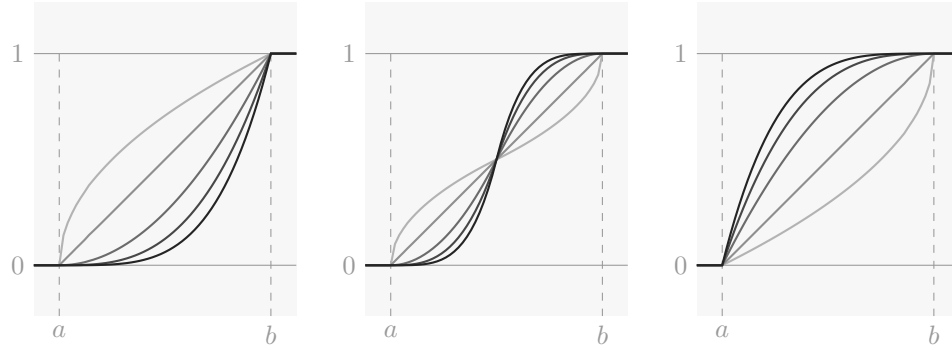


#### 4.1.2 The pow shape and friends (linear, quad, cubic, quart, and quint)

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

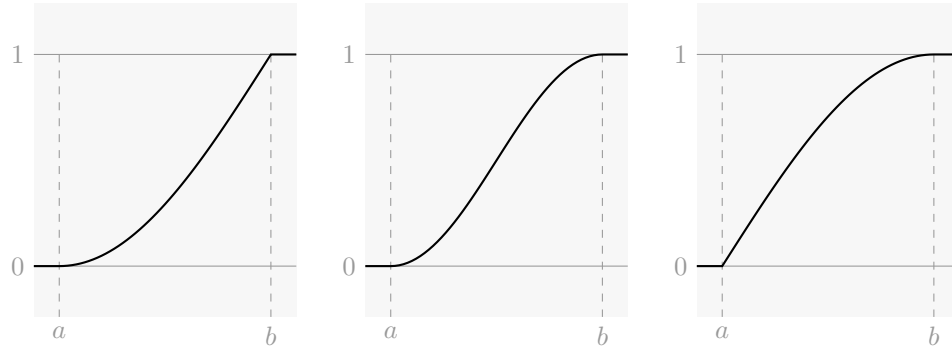
When  $n = 1$ , the function is linear between 0 and 1. For  $0 < n \leq 1$ , the ease-in form has discontinuous derivative at 0.

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 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  $0 \leq t \leq 1$ .

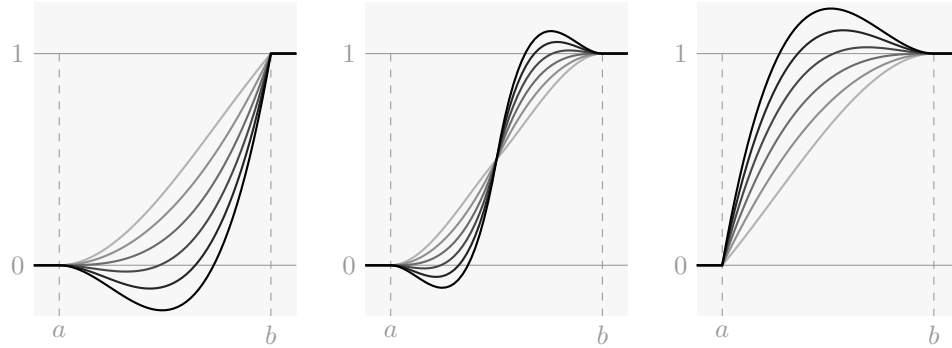


## 4.2 Other

### 4.2.1 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 by the PGF key `/easing/back/overshoot`. The parameter  $n$  defaults to 1.6.

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



## 5 Implementation

`\ifeasing@withfpu` This library uses  $\text{\TeX}$  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@withfputrue
6 \fi
7 \ifeasing@withfpu
8 \let\easing@divide\pgfmath@basic@divide@
9 \let\easing@cos\pgfmath@basic@cos@
10 \let\easing@exp\pgfmath@basic@exp@
11 \let\easing@ln\pgfmath@basic@ln@
12 \else
13 \let\easing@divide\pgfmathdivide@
14 \let\easing@cos\pgfmathcos@
15 \let\easing@exp\pgfmathexp@
16 \let\easing@ln\pgfmathln@
17 \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.

```

18 \def\easing@linearstep@ne#1{%
19   \begingroup
20   \pgf@x#1pt
21   \ifdim1pt<\pgf@x\pgf@x 1pt\fi
22   \ifdim0pt>\pgf@x\pgf@x 0pt\fi
23   \pgfmathreturn\pgf@x
24   \endgroup
25 }%
26 \expandafter\def
27 \csname easing@linearstep\ifeasing@withfpu @fixed\fi\endcsname#1#2#3{%
28   \begingroup
29   \pgf@xa#3pt
30   \pgf@xb#2pt
31   \pgf@xc#1pt
32   \ifdim\pgf@xb=\pgf@xc
33     \edef\pgfmathresult{\ifdim\pgf@xa>\pgf@xb 1\else 0\fi}%
34   \else
35     \advance\pgf@xa-\pgf@xc
36     \advance\pgf@xb-\pgf@xc
37     \easing@divide{\pgfmath@tonumber\pgf@xa}{\pgfmath@tonumber\pgf@xb}%
38     \easing@linearstep@ne\pgfmathresult
39     \fi
40     \pgfmathsmuggle\pgfmathresult
41   \endgroup
42 }%
43 \ifeasing@withfpu
44 \def\easing@linearstep@float#1#2#3{%
45   \begingroup
46   \pgfmathfloatsubtract{#3}{#1}%
47   \edef\pgf@tempa{\pgfmathresult}%
48   \pgfmathfloatsubtract{#2}{#1}%
49   \edef\pgf@tempb{\pgfmathresult}%
50   \pgfmathfloatifflags{\pgf@tempb}{0}{%
51     \pgfmathfloatifflags{\pgf@tempa}{-}{%
52       \edef\pgfmathresult{0}%
53     }{%
54       \edef\pgfmathresult{1}%
55     }%
56   }{%
57     \pgfmathfloatdivide\pgf@tempa\pgf@tempb
58     \pgfmathfloattofixed{\pgfmathresult}%
59     \easing@linearstep@ne\pgfmathresult
60   }%
61   \pgfmathsmuggle\pgfmathresult
62   \endgroup
63 }%
64 \def\easing@linearstep#1#2#3{%

```

```

65 \pgflibraryfpuiactive{%
66   \easing@linearstep@float{#1}{#2}{#3}}{%
67   \easing@linearstep@fixed{#1}{#2}{#3}}%
68 }%
69 \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.

```

70 \let\easing@lineareasein\easing@linearstep
71 \pgfmathdeclarefunction{lineareasein}{3}{%
72   \easing@lineareasein{#1}{#2}{#3}}%
73 \let\easing@lineareaseout\easing@linearstep
74 \pgfmathdeclarefunction{lineareaseout}{3}{%
75   \easing@lineareasein{#1}{#2}{#3}}%

```

`\easing@derive@easein@nefromstep@ne` The pattern in general is that, for each shape, we define the one-parameter version  
`\easing@derive@easeout@nefromstep@ne` of the step, ease-in, and ease-out routines interpolating between values 0 at 1 at  
`\easing@derive@step@nefromeasein@ne` the ends of the unit interval. Then by composing with `\easing@linearstep`, we  
`\easing@derive@easeout@nefromeasein@ne` 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.

```

76 \def\easing@derive@easein@nefromstep@ne#1{%
77   \expandafter\def\csname easing@#1easein@ne\endcsname##1{%
78     \begingroup
79     \pgf@x##1 pt
80     \divide\pgf@x 2
81     \csname easing@#1step@ne\endcsname{\pgfmath@tonumber\pgf@x}%
82     \pgf@x\pgfmathresult pt
83     \multiply\pgf@x 2
84     \pgfmathreturn\pgf@x
85   \endgroup
86 }%
87 }%
88 \def\easing@derive@easeout@nefromstep@ne#1{%
89   \expandafter\def\csname easing@#1easeout@ne\endcsname##1{%
90     \begingroup
91     \pgf@x##1 pt
92     \divide\pgf@x 2
93     \advance\pgf@x 0.5pt
94     \csname easing@#1step@ne\endcsname{\pgfmath@tonumber\pgf@x}%
95     \pgf@x\pgfmathresult pt
96     \multiply\pgf@x 2
97     \advance\pgf@x -1pt
98     \pgfmathreturn\pgf@x

```



```

99     \endgroup
100 }%
101 }%
102 \def\easing@derive@step@nefromeasein@ne#1{%
103   \expandafter\def\csname easing@#1step@ne\endcsname##1{%
104     \begingroup
105       \pgf@x##1 pt
106       \multiply\pgf@x 2
107       \ifdim\pgf@x<1pt
108         \csname easing@#1easein@ne\endcsname{\pgfmath@tonumber\pgf@x}%
109         \pgf@x\pgfmathresult pt
110         \divide\pgf@x 2
111       \else
112         \multiply\pgf@x -1
113         \advance\pgf@x 2pt
114         \csname easing@#1easein@ne\endcsname{\pgfmath@tonumber\pgf@x}%
115         \pgf@x\pgfmathresult pt
116         \divide\pgf@x 2
117         \multiply\pgf@x -1
118         \advance\pgf@x 1pt
119       \fi
120       \pgfmathreturn\pgf@x
121     \endgroup
122   }%
123 }%
124 \def\easing@derive@easeout@nefromeasein@ne#1{%
125   \expandafter\def\csname easing@#1easeout@ne\endcsname##1{%
126     \begingroup
127       \pgf@x##1pt
128       \multiply\pgf@x -1
129       \advance\pgf@x 1pt
130       \csname easing@#1easein@ne\endcsname{\pgfmath@tonumber\pgf@x}%
131       \pgf@x\pgfmathresult pt
132       \multiply\pgf@x -1
133       \advance\pgf@x 1pt
134       \pgfmathreturn\pgf@x
135     \endgroup
136   }%
137 }

```

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

```

138 \def\easing@pgfmathinstall#1{%
139   \pgfmathdeclarefunction{#1step}{3}{%
140     \easing@linearstep{##1}{##2}{##3}%
141     \csname easing@#1step@ne\endcsname\pgfmathresult
142   }%
143   \pgfmathdeclarefunction{#1easein}{3}{%
144     \easing@linearstep{##1}{##2}{##3}%

```

```

145 \csname easing@#1easein@ne\endcsname\pgfmathresult
146 }%
147 \pgfmathdeclarefunction{#1easeout}{3}{%
148 \easing@linearstep{##1}{##2}{##3}%
149 \csname easing@#1easeout@ne\endcsname\pgfmathresult
150 }%
151 }%

```

```

\begin{tikzpicture}
\draw[smooth] (0,0) .. (1,1);
\end{tikzpicture}

```

The smooth shape.

```

\def\easing@smoothstep@ne#1{%
\beginpgfgroup
\pgf@x#1pt
\edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
\multiply\pgf@x-2
\advance\pgf@x 3pt
\pgf@x\pgf@temp\pgf@x
\pgf@x\pgf@temp\pgf@x
\pgfmathreturn\pgf@x
\endpgfgroup
}%
\def\easing@derive@easein@nefromstep@ne{smooth}%
\def\easing@derive@easeout@nefromstep@ne{smooth}%
\def\easing@pgfmathinstall{smooth}%

```

```

\begin{tikzpicture}
\draw[smoothstep] (0,0) .. (1,1);
\end{tikzpicture}

```

The smoother shape.

```

\def\easing@smootherstep@ne#1{%
\beginpgfgroup
\pgf@x#1pt
\edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
\multiply\pgf@x 6
\advance\pgf@x -15pt
\pgf@x\pgf@temp\pgf@x
\advance\pgf@x 10pt
\pgf@x\pgf@temp\pgf@x
\pgf@x\pgf@temp\pgf@x
\pgf@x\pgf@temp\pgf@x
\pgfmathreturn\pgf@x
\endpgfgroup
}%
\def\easing@derive@easein@nefromstep@ne{smoother}%
\def\easing@derive@easeout@nefromstep@ne{smoother}%
\def\easing@pgfmathinstall{smoother}%

```

```

\begin{tikzpicture}
\draw[sine] (0,0) .. (1,1);
\end{tikzpicture}

```

The sine shape.

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-`.

```

183 \def\easing@sineeasein@ne#1{%
184   \beginpgfgroup
185   \pgf@x#1pt
186   \multiply\pgf@x 90
187   \easing@cos{\pgfmath@tonumber\pgf@x}%
188   \pgf@x\pgfmathresult pt
189   \multiply\pgf@x -1
190   \advance\pgf@x 1pt
191   \pgfmathreturn\pgf@x
192   \endpgfgroup
193 }%
194 \def\easing@sinestep@ne#1{%
195   \beginpgfgroup
196   \pgf@x#1pt
197   \multiply\pgf@x 180
198   \easing@cos{\pgfmath@tonumber\pgf@x}%
199   \pgf@x\pgfmathresult pt
200   \divide\pgf@x 2
201   \multiply\pgf@x -1
202   \advance\pgf@x 0.5pt
203   \pgfmathreturn\pgf@x
204   \endpgfgroup
205 }%
206 \easing@derive@easeout@nefromeasein@ne{sine}%
207 \easing@pgfmathinstall{sine}%

```

\easing@powstep@ne The pow shape.

\easing@poweasein@ne Because of some wonkiness in the FPU, instead of invoking the pow function from pgfmath, we compute  $t^n$  approximately by computing  $e^{n \ln t}$  using ln and exp instead (which is what pgfmath does anyway when the exponent is not an integer.)

\easing@poweaseout@ne

```

208 \pgfkeys{/easing/.is family}%
209 \pgfkeys{easing,
210   pow/exponent/.estore in=\easing@param@pow@exponent,
211   pow/exponent/.default=2.4,
212   pow/exponent}%
213 \def\easing@poweasein@ne#1{%
214   \beginpgfgroup
215   \pgf@x#1pt
216   \ifdim\pgf@x=0pt
217   \edef\pgfmathresult{0}%
218   \else
219   \easing@ln{#1}%
220   \pgf@x\pgfmathresult pt
221   \pgf@x\easing@param@pow@exponent\pgf@x
222   \easing@exp{\pgfmath@tonumber\pgf@x}%
223   \fi
224   \pgfmathsmuggle\pgfmathresult

```

```

225 \endgroup
226 }%
227 \easing@derive@easeout@nefromeasein@ne{pow}%
228 \easing@derive@step@nefromeasein@ne{pow}%
229 \easing@pgfmathinstall{pow}%

\.easing@quadstep@ne The quad-, cubic-, quart-, and quint- routines have explicit definitions.
\.easing@quadeasein@ne
\.easing@quadeaseout@ne 230 \def\easing@quadeasein@ne#1{%
\.easing@cubicstep@ne 231 \begingroup
\.easing@cubiceasein@ne 232 \pgf@x#1pt
\.easing@cubiceaseout@ne 233 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
\.easing@quartstep@ne 234 \pgf@x\pgf@temp\pgf@x
\.easing@quarteasein@ne 235 \pgfmathreturn\pgf@x
\.easing@quarteaseout@ne 236 \endgroup
\.easing@quintstep@ne 237 }%
\.easing@quinteasein@ne 238 \easing@derive@step@nefromeasein@ne{quad}%
\.easing@quinteaseout@ne 239 \easing@derive@easeout@nefromeasein@ne{quad}%
\.easing@quinteaseout@ne 240 \easing@pgfmathinstall{quad}%
241
242 \def\easing@cubiceasein@ne#1{%
243 \begingroup
244 \pgf@x#1pt
245 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
246 \pgf@x\pgf@temp\pgf@x
247 \pgf@x\pgf@temp\pgf@x
248 \pgfmathreturn\pgf@x
249 \endgroup
250 }%
251 \easing@derive@step@nefromeasein@ne{cubic}%
252 \easing@derive@easeout@nefromeasein@ne{cubic}%
253 \easing@pgfmathinstall{cubic}%
254
255 \def\easing@quarteasein@ne#1{%
256 \begingroup
257 \pgf@x#1pt
258 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
259 \pgf@x\pgf@temp\pgf@x
260 \pgf@x\pgf@temp\pgf@x
261 \pgf@x\pgf@temp\pgf@x
262 \pgfmathreturn\pgf@x
263 \endgroup
264 }%
265 \easing@derive@step@nefromeasein@ne{quart}%
266 \easing@derive@easeout@nefromeasein@ne{quart}%
267 \easing@pgfmathinstall{quart}%
268
269 \def\easing@quinteasein@ne#1{%
270 \begingroup
271 \pgf@x#1pt

```

```

272 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
273 \pgf@x\pgf@temp\pgf@x
274 \pgf@x\pgf@temp\pgf@x
275 \pgf@x\pgf@temp\pgf@x
276 \pgf@x\pgf@temp\pgf@x
277 \pgfmathreturn\pgf@x
278 \endgroup
279 }%
280 \easing@derive@step@nefromeasein@ne{quint}%
281 \easing@derive@easeout@nefromeasein@ne{quint}%
282 \easing@pgfmathinstall{quint}%

\begin{tikzpicture}
\draw[blue] (0,0) .. controls (1,1) and (2,1) .. (3,0);
\draw[blue] (3,0) .. controls (4,1) and (5,1) .. (6,0);
\draw[blue] (6,0) .. controls (7,1) and (8,1) .. (9,0);
\draw[blue] (9,0) .. controls (10,1) and (11,1) .. (12,0);
\draw[blue] (12,0) .. controls (13,1) and (14,1) .. (15,0);
\draw[blue] (15,0) .. controls (16,1) and (17,1) .. (18,0);
\draw[blue] (18,0) .. controls (19,1) and (20,1) .. (21,0);
\draw[blue] (21,0) .. controls (22,1) and (23,1) .. (24,0);
\draw[blue] (24,0) .. controls (25,1) and (26,1) .. (27,0);
\draw[blue] (27,0) .. controls (28,1) and (29,1) .. (30,0);
\draw[blue] (30,0) .. controls (31,1) and (32,1) .. (33,0);
\draw[blue] (33,0) .. controls (34,1) and (35,1) .. (36,0);
\draw[blue] (36,0) .. controls (37,1) and (38,1) .. (39,0);
\draw[blue] (39,0) .. controls (40,1) and (41,1) .. (42,0);
\draw[blue] (42,0) .. controls (43,1) and (44,1) .. (45,0);
\draw[blue] (45,0) .. controls (46,1) and (47,1) .. (48,0);
\draw[blue] (48,0) .. controls (49,1) and (50,1) .. (51,0);
\draw[blue] (51,0) .. controls (52,1) and (53,1) .. (54,0);
\draw[blue] (54,0) .. controls (55,1) and (56,1) .. (57,0);
\draw[blue] (57,0) .. controls (58,1) and (59,1) .. (60,0);
\draw[blue] (60,0) .. controls (61,1) and (62,1) .. (63,0);
\draw[blue] (63,0) .. controls (64,1) and (65,1) .. (66,0);
\draw[blue] (66,0) .. controls (67,1) and (68,1) .. (69,0);
\draw[blue] (69,0) .. controls (70,1) and (71,1) .. (72,0);
\draw[blue] (72,0) .. controls (73,1) and (74,1) .. (75,0);
\draw[blue] (75,0) .. controls (76,1) and (77,1) .. (78,0);
\draw[blue] (78,0) .. controls (79,1) and (80,1) .. (81,0);
\draw[blue] (81,0) .. controls (82,1) and (83,1) .. (84,0);
\draw[blue] (84,0) .. controls (85,1) and (86,1) .. (87,0);
\draw[blue] (87,0) .. controls (88,1) and (89,1) .. (90,0);
\draw[blue] (90,0) .. controls (91,1) and (92,1) .. (93,0);
\draw[blue] (93,0) .. controls (94,1) and (95,1) .. (96,0);
\draw[blue] (96,0) .. controls (97,1) and (98,1) .. (99,0);
\draw[blue] (99,0) .. controls (100,1) and (101,1) .. (102,0);
\draw[blue] (102,0) .. controls (103,1) and (104,1) .. (105,0);
\draw[blue] (105,0) .. controls (106,1) and (107,1) .. (108,0);
\draw[blue] (108,0) .. controls (109,1) and (110,1) .. (111,0);
\draw[blue] (111,0) .. controls (112,1) and (113,1) .. (114,0);
\draw[blue] (114,0) .. controls (115,1) and (116,1) .. (117,0);
\draw[blue] (117,0) .. controls (118,1) and (119,1) .. (120,0);
\draw[blue] (120,0) .. controls (121,1) and (122,1) .. (123,0);
\draw[blue] (123,0) .. controls (124,1) and (125,1) .. (126,0);
\draw[blue] (126,0) .. controls (127,1) and (128,1) .. (129,0);
\draw[blue] (129,0) .. controls (130,1) and (131,1) .. (132,0);
\draw[blue] (132,0) .. controls (133,1) and (134,1) .. (135,0);
\draw[blue] (135,0) .. controls (136,1) and (137,1) .. (138,0);
\draw[blue] (138,0) .. controls (139,1) and (140,1) .. (141,0);
\draw[blue] (141,0) .. controls (142,1) and (143,1) .. (144,0);
\draw[blue] (144,0) .. controls (145,1) and (146,1) .. (147,0);
\draw[blue] (147,0) .. controls (148,1) and (149,1) .. (150,0);
\draw[blue] (150,0) .. controls (151,1) and (152,1) .. (153,0);
\draw[blue] (153,0) .. controls (154,1) and (155,1) .. (156,0);
\draw[blue] (156,0) .. controls (157,1) and (158,1) .. (159,0);
\draw[blue] (159,0) .. controls (160,1) and (161,1) .. (162,0);
\draw[blue] (162,0) .. controls (163,1) and (164,1) .. (165,0);
\draw[blue] (165,0) .. controls (166,1) and (167,1) .. (168,0);
\draw[blue] (168,0) .. controls (169,1) and (170,1) .. (171,0);
\draw[blue] (171,0) .. controls (172,1) and (173,1) .. (174,0);
\draw[blue] (174,0) .. controls (175,1) and (176,1) .. (177,0);
\draw[blue] (177,0) .. controls (178,1) and (179,1) .. (180,0);
\draw[blue] (180,0) .. controls (181,1) and (182,1) .. (183,0);
\draw[blue] (183,0) .. controls (184,1) and (185,1) .. (186,0);
\draw[blue] (186,0) .. controls (187,1) and (188,1) .. (189,0);
\draw[blue] (189,0) .. controls (190,1) and (191,1) .. (192,0);
\draw[blue] (192,0) .. controls (193,1) and (194,1) .. (195,0);
\draw[blue] (195,0) .. controls (196,1) and (197,1) .. (198,0);
\draw[blue] (198,0) .. controls (199,1) and (200,1) .. (201,0);
\draw[blue] (201,0) .. controls (202,1) and (203,1) .. (204,0);
\draw[blue] (204,0) .. controls (205,1) and (206,1) .. (207,0);
\draw[blue] (207,0) .. controls (208,1) and (209,1) .. (210,0);
\draw[blue] (210,0) .. controls (211,1) and (212,1) .. (213,0);
\draw[blue] (213,0) .. controls (214,1) and (215,1) .. (216,0);
\draw[blue] (216,0) .. controls (217,1) and (218,1) .. (219,0);
\draw[blue] (219,0) .. controls (220,1) and (221,1) .. (222,0);
\draw[blue] (222,0) .. controls (223,1) and (224,1) .. (225,0);
\draw[blue] (225,0) .. controls (226,1) and (227,1) .. (228,0);
\draw[blue] (228,0) .. controls (229,1) and (230,1) .. (231,0);
\draw[blue] (231,0) .. controls (232,1) and (233,1) .. (234,0);
\draw[blue] (234,0) .. controls (235,1) and (236,1) .. (237,0);
\draw[blue] (237,0) .. controls (238,1) and (239,1) .. (240,0);
\draw[blue] (240,0) .. controls (241,1) and (242,1) .. (243,0);
\draw[blue] (243,0) .. controls (244,1) and (245,1) .. (246,0);
\draw[blue] (246,0) .. controls (247,1) and (248,1) .. (249,0);
\draw[blue] (249,0) .. controls (250,1) and (251,1) .. (252,0);
\draw[blue] (252,0) .. controls (253,1) and (254,1) .. (255,0);
\draw[blue] (255,0) .. controls (256,1) and (257,1) .. (258,0);
\draw[blue] (258,0) .. controls (259,1) and (260,1) .. (261,0);
\draw[blue] (261,0) .. controls (262,1) and (263,1) .. (264,0);
\draw[blue] (264,0) .. controls (265,1) and (266,1) .. (267,0);
\draw[blue] (267,0) .. controls (268,1) and (269,1) .. (270,0);
\draw[blue] (270,0) .. controls (271,1) and (272,1) .. (273,0);
\draw[blue] (273,0) .. controls (274,1) and (275,1) .. (276,0);
\draw[blue] (276,0) .. controls (277,1) and (278,1) .. (279,0);
\draw[blue] (279,0) .. controls (280,1) and (281,1) .. (282,0);
\draw[blue] (282,0) .. controls (283,1) and (284,1) .. (285,0);
\draw[blue] (285,0) .. controls (286,1) and (287,1) .. (288,0);
\draw[blue] (288,0) .. controls (289,1) and (290,1) .. (291,0);
\draw[blue] (291,0) .. controls (292,1) and (293,1) .. (294,0);
\draw[blue] (294,0) .. controls (295,1) and (296,1) .. (297,0);
\draw[blue] (297,0) .. controls (298,1) and (299,1) .. (300,0);
\draw[blue] (300,0) .. controls (301,1) and (302,1) .. (303,0);
\draw[blue] (303,0) .. controls (304,1) and (305,1) .. (306,0);
\draw[blue] (306,0) .. controls (307,1) and (308,1) .. (309,0);
\draw[blue] (309,0) .. controls (310,1) and (311,1) .. (312,0);
\draw[blue] (312,0) .. controls (313,1) and (314,1) .. (315,0);
\draw[blue] (315,0) .. controls (316,1) and (317,1) .. (318,0);
\draw[blue] (318,0) .. controls (
```

```

\easing@backstep@ne
\ easing@backeasein@ne
\ easing@backeaseout@ne
283 \pgfkeys{easing,
284   back/overshoot/.estore in=\easing@param@back@overshoot,
285   back/overshoot/.default=1.6,
286   back/overshoot}%
287 \def\easing@backeasein@ne#1{%
288   \beginpgfgroup
289   \pgf@x#1pt
290   \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
291   \advance\pgf@x -1pt
292   \pgf@x\easing@param@back@overshoot\pgf@x
293   \advance\pgf@x\pgf@temp pt
294   \pgf@x\pgf@temp\pgf@x
295   \pgf@x\pgf@temp\pgf@x
296   \pgfmathreturn\pgf@x
297   \endpgfgroup
298 }%
299 \easing@derive@step@nefromeasein@ne{back}%
300 \easing@derive@easeout@nefromeasein@ne{back}%
301 \easing@pgfmathinstall{back}%

```