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 2021/07/18. 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:

\begin{verbatim}
\input pgf
\usepgflibrary{easing}
\end{verbatim}

with \LaTeX{}:

\begin{verbatim}
\usepackage{pgf}
\usepgflibrary{easing}
\end{verbatim}

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 \texttt{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 \texttt{\pgfmathdeclarefunction}, so that they are recognised by \texttt{\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 \texttt{smoothstep(a,b,x)} against the argument \texttt{x}, with one endpoint \texttt{a = 0} and the other endpoint \texttt{b} ranging through the integers \(-1\) to \(3\):
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 \texttt{easing} are also available as “public” macros, such as \texttt{pgfmathsmoothstep}:

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

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 \textit{ease-in} form \texttt{⟨shape ⟩easein}(\(a, b, x\)) has easing applied near the endpoint \( a \).
- The \textit{ease-out} form \texttt{⟨shape ⟩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 \textit{step function} form \texttt{⟨shape ⟩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.
3.2 Specifying parameters

Some of these shapes can be modified by adjusting one or more parameters, which is done through \texttt{pgfkeys}: the parameter (\texttt{param}) for functions of shape (\texttt{shape}) is specified by setting the PGF key /easing/(\texttt{shape})/(\texttt{param}):
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 \( \frac{x-b}{b-a} \).

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 as \( 3t^2 - 2t^3 \) 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 as \( 10t^3 - 15t^4 + 6t^5 \) 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 \( n = 1, \ldots, 5 \), respectively. Computations for these shapes are implemented with Ti\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 \text{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 \texttt{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 \texttt{/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 \texttt{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.
This library uses TeX registers and PGF’s mathematical engine for computations. It is possible that the user is loading this library together with the floating point unit library. We save the basic routines from \texttt{pgfmath}, so that when this happens, the FPU doesn’t break everything when it does a switcharoo with the \texttt{pgfmath} macros.

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

If the FPU is loaded, \texttt{easing@linearstep} is instead named \texttt{easing@linearstep@fixed}, and we additionally define \texttt{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 \texttt{easing@linearstep} routine is the first step in the definition of all other routines that compute easing functions.
The linear ease-in and ease-out functions are identical to the linear step function. We define the respective macros so as not to surprise the user with their absence.
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 \texttt{\textbackslash 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 \texttt{\textbackslash easing@derive-from-} macros.

\begin{verbatim}
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} pt
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} pt
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} pt
111 \pgf@x\pgfmathresult pt
112 \divide\pgf@x 2
113 \endgroup
114 )%
115 )%
\end{verbatim}
\begin{verbatim}
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 }%
\end{verbatim}

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

\begin{verbatim}
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 }%
\end{verbatim}

The smooth shape.

\begin{verbatim}
154 \def\easing@smoothstep@ne#1{%
155 \begingroup
156 \endgroup
\end{verbatim}

\begin{verbatim}
158 \easing@smootheasein@ne
159 \easing@smootheaseout@ne
\end{verbatim}
The smoother shape.

Because of some wonkiness in the FPU, \texttt{pgfmath@pow@basic@} actually doesn't work. Instead of invoking the \texttt{pow} function, we compute $t^n$ approximately by computing $e^{n \ln t}$ using \texttt{ln} and \texttt{exp} instead (which is what \texttt{pgfmath} does anyway when the exponent is not an integer.)
\begin{verbatim}
\def\easing@quadstepone{\edef\pgfmathresult{0.25}\edef\pgfmathresult{0.5}\edef\pgfmathresult{0.75}}\def\easing@quadeaseinone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@quadeaseoutone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@cubiceaseinone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@cubiceaseoutone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@quartstepone{\edef\pgfmathresult{0.25\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.75\pgfmathresult}}\def\easing@quarteaseinone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@quarteaseoutone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@quintstepone{\edef\pgfmathresult{0.25\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.75\pgfmathresult}}\def\easing@quinteaseinone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}\def\easing@quinteaseoutone#1{\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}\edef\pgfmathresult{0.5\pgfmathresult}}
\end{verbatim}

The \texttt{quad}–, \texttt{cubic}–, \texttt{quart}–, and \texttt{quint}– routines have explicit definitions.
The back shape.

We write down both the ease in and step forms of this, since they are simple compared to what would have been obtained by \texttt{\easing@derive}–.

\begin{verbatim}
\def\easing@backeasein@ne#1{%
  \begingroup
  \pgf@x#1pt
  \edef\pgf@temp{\pgfmath@tonumber\pgf@x}%
  \pgf@x\pgf@temp\pgf@x
  \pgf@x\pgf@temp\pgf@x
  \pgf@x\pgf@temp\pgf@x
  \pgf@x\pgf@temp\pgf@x
  \advance\pgf@x -1pt
  \pgf@x\easing@param@back@overshoot\pgf@x
  \advance\pgf@x\pgf@temp pt
  \pgfmathreturn\pgf@x
  \endgroup
%
\end{verbatim}

The sine shape.

\begin{verbatim}
\def\easing@sineeasein@ne#1{%
  \begingroup
  \pgf@x#1pt
  \multiply\pgf@x 90
\end{verbatim}
\begin{verbatim}
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 \begingroup
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}
304 \easing@expstep@ne
305 \easing@expeasein@ne
306 \easing@expeaseout@ne
307 \pgfkeys{easing,
308 exp/speed/.estore in=\easing@param@exponent@speed,
309 exp/speed/.default=7.2,
310 \def\easing@expeasein@ne#1{%
311 \begingroup
312 \pgf@x#1pt
313 \advance\pgf@x -1pt
314 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}\
315 \pgf@x\pgf@temp\pgf@x
316 \multiply\pgf@x -1
317 \pgfmathsmuggle\pgfmathresult
318 \endgroup
319 }%
320 \easing@derive@step@nefromeasein@ne{exp}
321 \easing@derive@easeout@nefromeasein@ne{exp}
322 \easing@pgfmathinstall{exp}
323 \easing@circstep@ne
324 \easing@circeasein@ne
325 \easing@circeaseout@ne
326 \pgfkeys{easing,
327 circ/speed/.estore in=\easing@param@exponent@speed,
328 circ/speed/.default=7.2,
329 circ/speed}
330 \def\easing@circeasein@ne#1{%
331 \begingroup
332 \pgf@x#1pt
333 \advance\pgf@x -1pt
334 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}\
335 \pgf@x\pgf@temp\pgf@x
336 \multiply\pgf@x -1
337 \edef\pgf@temp{\pgfmath@tonumber\pgf@x}\
338 \pgf@x\pgf@temp\pgf@x
339 \pgfmathsmuggle\pgfmathresult
340 \endgroup
341 }%
342 \easing@derive@step@nefromeasein@ne{circ}
343 \easing@derive@easeout@nefromeasein@ne{circ}
344 \easing@pgfmathinstall{circ}
\end{verbatim}

The \texttt{exp} shape.

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

The \texttt{circ} shape.
The elastic shape.

\easing@elasticstepone
\easing@elasticeaseinone
\easing@elasticeaseoutone

\pgfkeys{easing,
  elastic/frequency/.estore in=\easing@param@elastic@frequency, elastic/damping/.estore in=\easing@param@elastic@damping, elastic/frequency/.default=3, elastic/damping/.default=7.2, elastic/frequency, elastic/damping)}
def\easing@elasticeaseinone#1{% begingroup \pgf@xa#1pt \advance\pgf@xa -1pt \pgf@xb-\pgf@xa \pgf@xa\easing@param@elastic@damping\pgf@xa \easing@exp\pgfmath@tonumber\pgf@xa\pgf@xa\pgfmathresult pt \pgf@xb 360\pgf@xb \pgf@xb\easing@param@elastic@frequency\pgf@xb \easing@cos\pgfmath@tonumber\pgf@xb\pgf@xa \pgfmathreturn\pgf@xa \endgroup \easing@derive@step@onefromeasein@one{elastic} \easing@derive@easeout@onefromeasein@one{elastic} \easing@pgfmathinstall{elastic}

Change History

0.1
  General: Initial version ........ 1