Easing Expressions in After Effects

Juan José Idrovo Macías

April 27, 2025

1 Introduction

Often when working in After Effects, it is easier to use expressions rather than keyframes, especially when dealing with motion graphics. A common example would be the *noise*() expression, very similar to *wiggle*(), which adds a random range of noise to a parameter. An applied example would be adding *noise*(*time*) to the x and y position of a layer to create a vibration or camera shake for the entire duration of the layer. This is much easier than keyframing the movement by hand.

The catch is that without keyframes, there is no obvious way to ease the effect in and out. It is stuck vibrating for the entire duration of the layer. There is a workaround to this, by getting clever with sliders pick-whipped to variables, but it only pushes the problem back a level. You would still need to manually keyframe the sliders. Albeit, its a simpler task than hand keyframing the property itself, but I consider it cheating. The goal is to fully animate a property using only expressions.

We will use the *noise*() expression as the primary effect example, so if you are unfamiliar with how the expression works, I highly recommend you click here.

This paper will explore the linear, logistic, and cubic easing of expressions. The general framework of our After Effects code will have the form

$$value + \text{effect}()f(t)$$

where *value* is the default or initial condition of the property, effect() is the effect we want to apply, and $f(t)$ is our easing function as a function of time.

2 Linear Ease

2.1 Deriving the Formula

Let $f(t)$ be a function such that $f(a)$ and $f(b)$ are the starting and ending strength of an effect() at $t = a$ and $t = b$, respectively, where $a \neq b$.

To find a linear fit for the two points, we can use point slope form

$$f(t) - f(a) = \frac{f(b) - f(a)}{b - a}\ (t - a)$$

Rearranging

$$\begin{aligned} f(t) &= \frac{f(b) - f(a)}{b - a}\ (t - a) + f(a) \\ &= \frac{f(b) - f(a)}{b - a}\ t - \frac{f(b) - f(a)}{b - a}\ a + \frac{b - a}{b - a}\ f(a) \\ &= \frac{f(b)t - f(a)t - f(b)a + f(a)a + f(a)b - f(a)a}{b - a} \\ &= \frac{f(b)t - f(a)t - f(b)a + f(a)b}{b - a} \end{aligned}$$

Factoring out $f(a)$ and $f(b)$

$$f(t) = \frac{f(b)(t - a) + f(a)(b - t)}{b - a}\ , \qquad \text{General Form}$$

It should also be noted that by factoring out t instead, we get

$$f(t) = \frac{f(b) - f(a)}{b - a}\ t + \frac{f(a)b - f(b)a}{b - a}\ , \qquad \text{Slope-Intercept Form}$$

Although both forms are valid, we will use the General Form because it is easier to work with fewer terms in After Effects. Now, let us apply this formula to a concrete example.

2.2 Example 1 - General Case

The following example will be used multiple times in this paper.

Suppose we want to add camera shake via *noise*() to the position property of a 24fps layer such that it starts off shaking with a maximum deviation of 10 pixels. Then, at exactly 3

seconds and 4 frames in, linearly increase the shake until exactly 6 seconds and 22 frames in, where it will be maximum deviation of 125 pixels for the rest of the layer.

Recall

$$value + \text{effect}()f(t)$$

Something convenient about the $noise()$ expression is that it outputs values between $[-1, 1]$. Therefore multiplying by $f(t)$ will act as a scalar. With this in mind, we can code the following

$$\begin{aligned} t &= time * 24 \\ a &= 3 * 24 + 4 \\ b &= 6 * 24 + 22 \\ f(a) &= 10 \\ f(b) &= 125 \end{aligned}$$

if $(t < a)$

$$value + [noise(t),\ noise(t + 12345)] * f(a)$$

else if $(a <= t\ \&\&\ t <= b)$

$$value + [noise(t),\ noise(t + 12345)] * [\frac{f(b)(t - a) + f(a)(b - t)}{b - a}\ ]$$

else if $(t > b)$

$$value + [noise(t),\ noise(t + 12345)] * f(b)$$

And just like that, we are done.

One thing to note is that After Effects will output an error for every poorly defined function. It is important to change the notation and replace $f(a)$ with A, and $f(b)$ with B. Here is the corrected After Effects code in plain-text for easy copying:

```
t = time*24
a = 3*24 + 4
b = 6*24 + 22
A = 10
B = 125

if (t < a)
    value + [noise(t), noise(t+12345)] * A
else if (a <= t && t <= b)
    value + [noise(t), noise(t+12345)] * [(B*(t-a)+A*(b-t))/(b-a)]
else if (t > b)
    value + [noise(t), noise(t+12345)] * B
```

For the purpose of visual clarity, we will continue to use the standard mathematical notation.

2.3 Example 2 - Unit Scalar

In the previous example, we used the fact that $noise()$ outputs values between $[-1, 1]$ to our advantage. This made made satisfying the pixel deviation easy by simply setting those values to $f(a)$ and $f(b)$. However in some cases, we will already be satisfied with the intensity of the effect, and just need to ease it in from 0 to 100 percent.

For this example, let us use the same timing for a and b, and embed the desired ending deviation of 125 pixels into the $noise()$.

Plugging in $f(a) = 0$, $f(b) = 1$ to the General Form

$$f(t) = \frac{(1)(t-a) + (0)(b-t)}{b-a}$$
$$f(t) = \frac{t-a}{b-a}\,, \qquad \text{Unit Scalar}$$

Then the remaining code reduces to

$$t = time * 24$$
$$a = 3 * 24 + 4$$
$$b = 6 * 24 + 22$$

if $(t < a)$

$$value$$

else if $(a <= t \;\&\&\; t <= b)$

$$value + [[noise(t),\ noise(t + 12345)] * 125] * (\frac{t - a}{b - a}\ )$$

else if $(t > b)$

$$value + [[noise(t),\ noise(t + 12345)] * 125]$$

Although the General Form could have achieved the same effect, the Unit Scalar simplifies the code, making it easier to work with under certain conditions.

```
t = time*24
a = 3*24 + 4
b = 6*24 + 22

if (t < a)
    value
else if (a <= t && t <= b)
    value + [noise(t), noise(t+12345)]*125 * [(t-a)/(b-a)]
else if (t > b)
    value + [noise(t), noise(t+12345)]*125
```

3 Logistic Ease

3.1 Overview

The logistic function is defined as follows

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} + C$$

where L is the upper bound, k is the growth rate, x_0 is the x value of the midpoint, and C is the lower bound.

Since f is already simplified, we can directly substitute

$$f(t) = \frac{f(b)}{1 + e^{-k(t-\frac{a+b}{2})}} + f(a), \qquad \text{General Form}$$
$$f(t) = \frac{1}{1 + e^{-k(t-\frac{a+b}{2})}}, \qquad \text{Unit Scalar}$$

The only new variable in the equation is the growth rate, k. In the linear case, our growth rate was the slope between the points $(a, f(a))$, and $(b, f(b))$. That is, the growth rate was dependent on our defined variables. In the logistic case however, k is independent. This means that we could choose an arbitrary k, and the function would still technically hold.

The issue comes with choosing the "right" k. For example, if we chose an arbitrary large k such that the resulting curve is incredibly steep around the midpoint (resembling a step function), then the transition from $(a, f(a))$, to $(b, f(b))$ would not appear smooth, since the function is only visibly growing in the immediate neighborhood of the midpoint, and is virtually flat everywhere else. Conversely, if we choose an arbitrary small k, the resulting curve would be so flat (resembling the line $f(t) = f(\frac{a+b}{2})$, that there would not be a meaningful transition between $(a, f(a))$, and $(b, f(b))$), as well as a glaring discontinuity to the left of $(a, f(a))$, and to the right of $(b, f(b))$.

Now, we could open up the graph editor and adjust k until the curve looks reasonable, but we are mathematicians, so let us define what "looking reasonable" actually entails.

3.2 Finding a suitable growth rate

Let us say that a suitable growth rate, k, will satisfy $f'(a) = 0.05$. This way, the logistic function will be reasonably flat around $(a, f(a))$, and have a reasonably imperceptible discontinuity. Since the derivative is symmetric along the midpoint, then satisfying the condition will also satisfy $f'(b) = 0.05$, and the same desired properties at $(b, f(b))$.

Because of the condition $f'(a) = 0.05$, it will not make sense in the following math for f(a) to be our lower bound, since we are truncating the function on both ends. Therefore, when manipulating equations, we must set the lower bound to the constant C. Similarly, we must also set the upper bound to the constant L. These reflect the original logistic function.

Here is the derivative of the logistic function

$$f'(t) = \frac{k}{L} f(t)[L - f(t)]$$

Plugging in $f'(a) = 0.05$, and solving for k

$$0.05 = \frac{k}{L} f(a)[L - f(a)]$$
$$k = \frac{0.05L}{f(a)[L - f(a)]}$$

To solve for k in the equation above, we need to know the value of $f(a)$. We will use the logistic function.

$$f(a) = \frac{L}{1 + e^{-k(a - \frac{a+b}{2})}} + C$$
$$1 + e^{-k(\frac{a-b}{2})} = \frac{L}{f(a) - C}$$
$$-k\frac{a-b}{2} = ln|\frac{L}{f(a) - C} - 1|$$
$$k = \frac{-2}{a-b} ln|\frac{L}{f(a) - C} - 1|$$

Putting the two equations together, let us try to solve for $f(a)$

$$\frac{0.05L}{f(a)[L - f(a)]} = \frac{-2}{a-b} ln|\frac{L}{f(a) - C} - 1|$$
$$-0.05L\frac{a-b}{2} = f(a)[L - f(a)] ln|\frac{L}{f(a) - C} - 1|$$
$$-0.025L(a-b) = ln|(\frac{L}{f(a) - C} - 1)^{f(a)[L - f(a)]}|$$
$$e^{-0.025L(a-b)} = (\frac{L}{f(a) - C} - 1)^{f(a)[L - f(a)]}$$

We have arrived at a roadblock. The above equation is transcendental since it is in the general form of $f(x) = x^x$, where $x = f(a)$. This means that there is no algebraic way to isolate $f(a)$, and therefore no way to algebraically solve for k. Our only alternative is to use numerical methods in a scripting environment to approximate a solution to k.

Since we are not programmers, let us ask ChatGPT to write a Python script to find k.

```
import numpy as np
from scipy.optimize import fsolve
import matplotlib.pyplot as plt

# Constants
a = 3*24+4        # Start point
b = 6*24+22       # End point
B = 125       # Max value of the logistic function
x0 = (a+b)/2       # Midpoint
desired_slope = 0.05  # Desired slope at x = a

# Define slope of logistic function at x = a
def slope_at_zero(k):
    if k <= 0:
        return np.inf  # Avoid invalid values
    exp_term = np.exp(-k * (a - x0))
    return B * k * exp_term / (1 + exp_term)**2 - desired_slope

# Plot slope vs k to find multiple roots
k_vals = np.linspace(0.001, 2, 1000)
slope_vals = [B * k * np.exp(-k * (a - x0)) /
             (1 + np.exp(-k * (a - x0)))**2 for k in k_vals]

plt.figure(figsize=(8, 4))
plt.plot(k_vals, slope_vals, label="f'("+str(a)+") vs k")
plt.axhline(desired_slope, color='red',
            linestyle='--', label='Target slope = '+str(desired_slope))
plt.title("Slope at a="+str(a)+" vs Growth Rate k
          (B="+str(B)+", x0="+str(x0)+")")
plt.xlabel("k (growth rate)")
plt.ylabel("f'(0)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

# Solve for the non-trivial solution (visible S-curve)
nontrivial_guess = 0.1
k_nontrivial = fsolve(slope_at_zero, nontrivial_guess)[0]

print(f"Non-trivial solution: k ~ k_nontrivial:.6f")
```

Running this script will give us the "non-trivial solution" for k. It should be noted that since the numerical method is based on the Newton-Raphson method, it will sometimes be necessary to change the "nontrivial-guess" parameter to an x-value that is under the blue curve, and above the red line.

To check if the k value works, I created this Desmos visualizer, which by default has the givens from Example 1, and the k value from the Python script, 0.128055.

3.3 Example 3 - General Case

Now, let us return to After Effects with our new found k. We can drop the C and L naming convention, and revert to $f(a)$ and $f(b)$ respectively. The code will look structurally identical to Example 1.

$$\begin{aligned} t &= time * 24 \\ a &= 3 * 24 + 4 \\ b &= 6 * 24 + 22 \\ f(a) &= 10 \\ f(b) &= 125 \\ k &= 0.128055 \end{aligned}$$

if $(t < a)$

$$value + [noise(t),\ noise(t + 12345)] * f(a)$$

else if $(a <= t\ \&\&\ t <= b)$

$$value + [noise(t),\ noise(t + 12345)] * [\frac{f(b)}{1 + e^{-k(t - \frac{a+b}{2})}}]$$

else if $(t > b)$

$$value + [noise(t),\ noise(t + 12345)] * f(b)$$

```
t = time*24
a = 3*24 + 4
b = 6*24 + 22
A = 10
B = 125
k = 0.128055

if (t < a)
    value + [noise(t), noise(t+12345)] * A
else if (a <= t && t <= b)
    value + [noise(t), noise(t+12345)] *
    [B/(1+Math.exp(-k*(t-[(a+b)/2])))]
else if (t > b)
    value + [noise(t), noise(t+12345)] * B
```

3.4 Example 4 - Unit Scalar

We could apply the code structure from Example 2 to the logistic case, and it would work. However recall that the logistic function has upper and lower asymptotes. This means that the code can be greatly reduced.

$$
\begin{aligned}
t &= time * 24 \\
a &= 3 * 24 + 4 \\
b &= 6 * 24 + 22 \\
f(a) &= 0 \\
f(b) &= 1 \\
k &= 0.034300
\end{aligned}
$$

$$value + [noise(t),\ noise(t + 12345)] * [\frac{f(b)}{1 + e^{-k(t - \frac{a+b}{2})}}]$$

4 Cubic Ease

4.1 Deriving the Formula

We want to construct a unit scalar cubic function that satisfies the following

$$f(a) = 0 \tag{1}$$
$$f(b) = 1 \tag{2}$$
$$f'(a) = 0 \tag{3}$$
$$f'(b) = 0 \tag{4}$$
$$f''(\frac{a+b}{2}) = 0 \tag{5}$$
$$f'''(t) = -1 \tag{6}$$

(1) and (2) establish that our function behaves like a unit scalar. (3) and (4) make sure that the function is smoothly easing in and out of the boundaries at a and b. (5) accounts for the inflection point, and adds symmetry at the midpoint to the transition. Finally, (6) ensures that the inflection point goes from concave up to down (increasing for $a \leq t \leq b$). It is also important to note that since the function has symmetry at the midpoint, then (3) $\iff$ (4).

Let us begin with (5) and work our way up

$$f'''(t) = -1$$
$$\int f'''(t)dt = \int (-1)dt$$
$$f''(t) = -t + c_1$$

Plugging in $f''(\frac{a+b}{2}) = 0$

$$0 = -\frac{a+b}{2} + c_1$$
$$c_1 = \frac{a+b}{2}$$

Substituting in c_1

$$f''(t)dt = -t + \frac{a+b}{2}$$
$$\int f''(t)dt = \int (-t + \frac{a+b}{2})dt$$
$$f'(t) = -\frac{1}{2}t^2 + \frac{a+b}{2}t + c_2$$

Plugging in $f'(a) = 0$

$$0 = -\frac{1}{2}a^2 + \frac{a+b}{2}a + c_2$$
$$c_2 = \frac{1}{2}a^2 - \frac{a+b}{2}a$$

Substituting in c_2

$$f'(t)dt = -\frac{1}{2}t^2 + \frac{a+b}{2}t + (\frac{1}{2}a^2 - \frac{a+b}{2}a)$$
$$\int f'(t)dt = \int(-\frac{1}{2}t^2 + \frac{a+b}{2}t + (\frac{1}{2}a^2 - \frac{a+b}{2}a))dt$$
$$f(t) = -\frac{1}{6}t^3 + \frac{a+b}{4}t^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)t + c_3$$

Plugging in f(a) = 0

$$0 = -\frac{1}{6}a^3 + \frac{a+b}{4}a^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)a + c_3$$
$$c_3 = \frac{1}{6}a^3 - \frac{a+b}{4}a^2 - (\frac{1}{2}a^2 - \frac{a+b}{2}a)a$$

Substituting in c_3

$$f(t) = -\frac{1}{6}t^3 + \frac{a+b}{4}t^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)t + (\frac{1}{6}a^3 - \frac{a+b}{4}a^2 - (\frac{1}{2}a^2 - \frac{a+b}{2}a)a)$$

Finally, to ensure $f(b) = 1$, we must scale down by $f(b)$

$$f(t) = \frac{-\frac{1}{6}t^3 + \frac{a+b}{4}t^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)t + (\frac{1}{6}a^3 - \frac{a+b}{4}a^2 - (\frac{1}{2}a^2 - \frac{a+b}{2}a)a)}{-\frac{1}{6}b^3 + \frac{a+b}{4}b^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)b + (\frac{1}{6}a^3 - \frac{a+b}{4}a^2 - (\frac{1}{2}a^2 - \frac{a+b}{2}a)a)}, \qquad \text{Unit Scalar}$$

And there is our final unit scalar cubic formula. To get the general form, we can scale by $f(b)$ and add $f(a)$

General Form

$$f(t) = f(b)\frac{-\frac{1}{6}t^3 + \frac{a+b}{4}t^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)t + (\frac{1}{6}a^3 - \frac{a+b}{4}a^2 - (\frac{1}{2}a^2 - \frac{a+b}{2}a)a)}{-\frac{1}{6}b^3 + \frac{a+b}{4}b^2 + (\frac{1}{2}a^2 - \frac{a+b}{2}a)b + (\frac{1}{6}a^3 - \frac{a+b}{4}a^2 - (\frac{1}{2}a^2 - \frac{a+b}{2}a)a)} + f(a)$$

5 Cosine Ease

Cosine Ease

$$f(t) = f(b)[-\frac{1}{2}\cos(\frac{t-a}{b-a}\pi) + \frac{1}{2}] + f(a)$$

Taylor Series centered at $t = \frac{a+b}{2}$

$$u = \frac{t-a}{b-a}\pi$$

$$f(t) = f(b)[-\frac{1}{2}((\frac{\pi}{2} - u) + \frac{(u-\frac{\pi}{2})^3}{6} - \frac{(u-\frac{\pi}{2})^5}{120} + \frac{(u-\frac{\pi}{2})^7}{5040}) + \frac{1}{2}] + f(a)$$

Arc-tangent

$$k = \frac{\frac{f(b)}{0.05\pi} + \sqrt{(\frac{-f(b)}{0.05\pi})^2 - (a-b)^2}}{\frac{(a-b)^2}{2}}$$

$$f(t) = f(b)[\frac{1}{\pi}\arctan(k(t - \frac{a+b}{2})) + \frac{1}{2}] + f(a)$$

This formula was derived in a similar fashion to the logistic. The growth rate at $t = a$ needed to be 0.05. However, $arctan(t)$ does not converge fast enough for a smooth transition. $f'(a) = 0.05$ still creates a noticeable discontinuity, and any values smaller makes f(t) resemble a step function.