New Functions for Blending

Tozoni, Davi Colli
davi.tozoni@nyu.edu

September 6, 2018

1 Original blending (exponential function)

The original blending function used in the `Microstructure` package was proposed in Panetta et al. [1]. The general idea is to merge two edges of the truss-like structure using a smooth approximation of the min function, which has the form shown below:

$$KS_\rho(y_1, \ldots, y_n) = -\frac{1}{\rho} \ln \left(\sum_{i=1}^{n} e^{-\rho y_i} \right) \tag{1}$$

where the only parameter is ρ, which is related to the size of the blending region.

Let's take a look in the version with only two values y_1 and y_2:

$$\begin{aligned} KS_\rho(y_1, y_2) &= -\frac{1}{\rho} \ln \left(e^{-\rho y_1} + e^{-\rho y_2} \right) \\ &= -\frac{1}{\rho} \ln \left(e^{-\rho \frac{(y_1+y_2)}{2}} \left(e^{-\rho \frac{y_1-y_2}{2}} + e^{\rho \frac{y_1-y_2}{2}} \right) \right) \\ &= \frac{(y_1+y_2)}{2} - \frac{1}{\rho} \ln \left(e^{-\rho \frac{y_1-y_2}{2}} + e^{\rho \frac{y_1-y_2}{2}} \right) \\ &= z - \frac{1}{\rho} \ln \left(e^{-\rho \epsilon} + e^{\rho \epsilon} \right) \end{aligned}$$

where $z = \frac{(y_1+y_2)}{2}$ and $\epsilon = \frac{y_1-y_2}{2}$.

Now, notice that the term $\frac{1}{\rho} \ln \left(e^{-\rho \epsilon} + e^{\rho \epsilon} \right)$ approximates the absolute function on ϵ. See Figure 1.

As said before, ρ controls the size of the region where blending occurs. If we study the smooth absolute function, we will see that the approximating region (where result is not similar to original function) is between $-\frac{2}{\rho}$ to $\frac{2}{\rho}$. The higher the value of ρ, the smaller is the error between approximation and absolute value.

This indicates that we could use a different smooth approximation to the absolute value here as base for our other blending functions. Basically, each of them chooses a different way, using polynomials, to approximate the absolute function. The polynomial coefficients are then the new parameters in the shape optimization.

2 Polynomial Blending (with monotonically increasing derivative)

The polynomial blending idea is to create a polynomial that has the exact same behavior of the absolute function outside interval $[-\frac{2}{\rho}, \frac{2}{\rho}]$ but is smooth inside the interval. The first idea was to

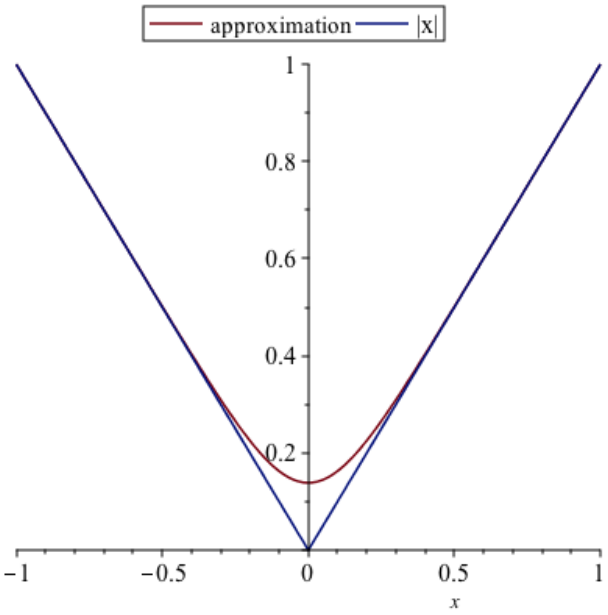

Figure 1: Comparison of absolute function and smooth version given by $\frac{1}{\rho}\ln\left(e^{-\rho\epsilon}+e^{\rho\epsilon}\right)$.

generate a function $f(x)$ with **monotonically increasing derivative**, to create a function that is **concave upward**.

First, let's study the derivative of the absolute function and the smooth approximation used in [1]. See Figure 2.

Now, notice that, to achieve our objective, we need a derivative that has positive slope during the whole approximation interval and which has derivative equal to 0 on both ends of the blending region. This defines the properties we want for the second derivative of the $f(x)$, called here $r(x)$. An example of a valid $r(x)$ is shown in Figure 3.

But how to construct function $f(x)$ based on this. Let's work on the opposite direction. First, let's create function $r(x)$ corresponding to the second derivative. To make sure it is always positive and 0 at $-\frac{2}{\rho}$ and $\frac{2}{\rho}$ we proposed the following:

$$r(x) = -\left(a_0 + a_2x^2 + a_4x^4 \cdots + a_{2n}x^{2n}\right)^2 \left(x+\frac{2}{\rho}\right)\left(x-\frac{2}{\rho}\right) \tag{2}$$

where $a_{2*(i-1)}$ corresponds to the ith parameter for blending (outside ρ). Notice that we removed the odd coefficients in order to make sure the function has reflectional symmetric with respect to the y axis.

Then, we build $h(x)$ by integrating $r(x)$, starting from $-\frac{2}{\rho}$:

$$h(x) = \int_{-2/\rho}^{x} r(z)dz$$

and we then normalize this integral so that it can be between -1 and 1, obtaining the slope function $g(x)$.

$$g(x) = 2\frac{h(x)}{h(2/\rho)} - 1$$

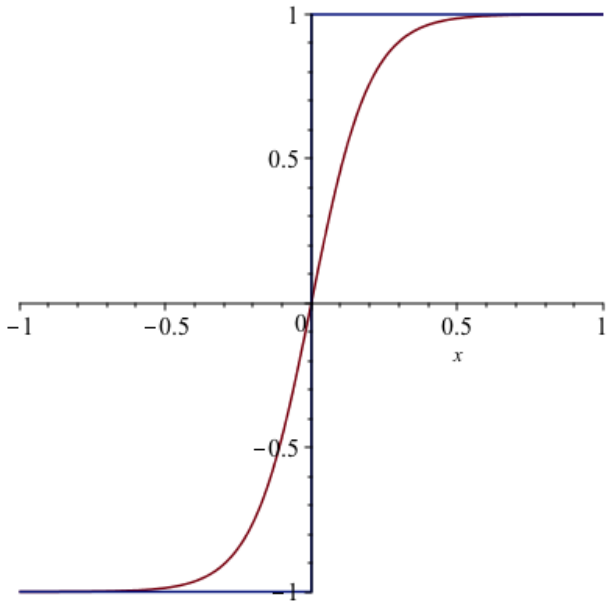

Figure 2: Comparison of derivative of absolute function and smooth version given by $\frac{1}{\rho}\ln\left(e^{-\rho\epsilon}+e^{\rho\epsilon}\right)$.

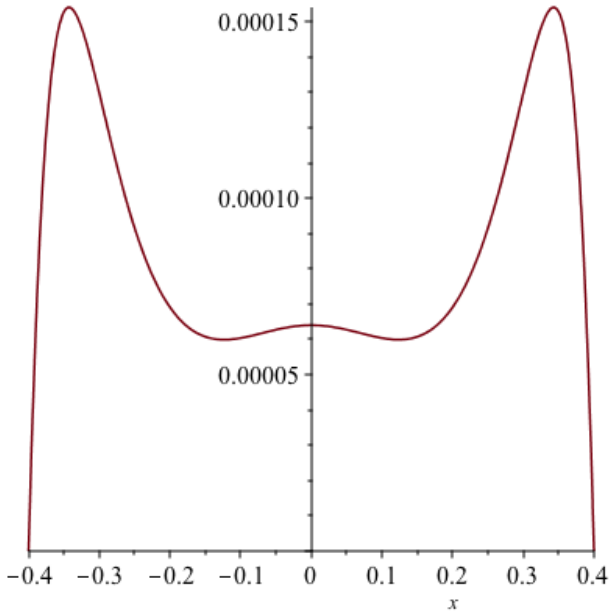

Figure 3: Example of function $r(x)$, representing second derivative of $f(x)$.

Again, to obtain now our smooth absolute function, we do the same thing, integrating $g(x)$:

$$q(x)=\int_{-2/\rho}^{x} g(z)dz$$

but we need to make sure that $f(2/\rho) = 2/\rho$. To obtain this, we need an offset a on $q(x)$ such that $f(2/\rho) = q(2/\rho) + a = 2/\rho$. So, $a = 2/\rho - q(2/\rho)$. And, finally,

$$f(x) = q(x) - 2/\rho - q(2/\rho)$$

An example of the final $f(x)$ is shown in Figure 4.

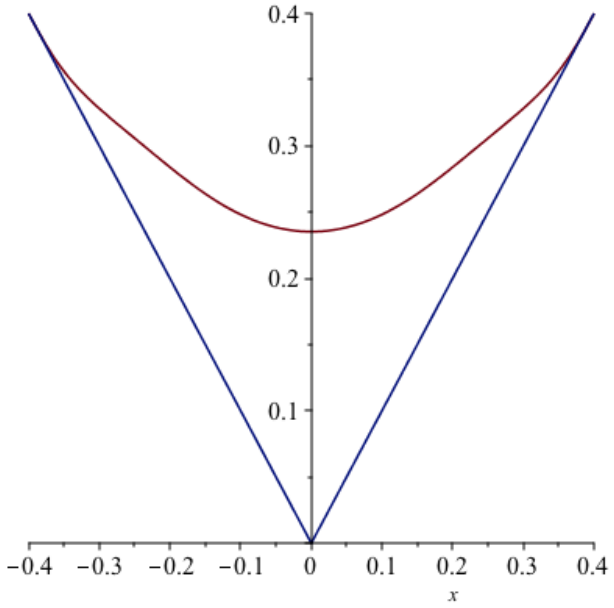

Figure 4: Example of function $f(x)$ with monotonically increasing derivative, generating convex blending shape.

To understand better the functions and to plot examples, please take a look on Maple script `KS_monotonic_derivative_8parameters.mw`.

3 Non-convex Polynomial Blending

After experiments, we noticed that in some cases it seemed important to have blending regions that are not always convex. To obtain this effect, we simply removed power of 2 from the first polynomial term in $r(x)$, obtaining:

$$r(x) = -\left(a_0 + a_2x^2 + a_4x^4 \cdots + a_{2n}x^{2n}\right)\left(x + \frac{2}{\rho}\right)\left(x - \frac{2}{\rho}\right) \tag{3}$$

The rest of the steps to obtain $f(x)$ are exactly the same as before. An example is shown in Figure 5. To find more information, please refer to script `KS_nonconvex_8parameters.mw`.

4 Piecewise Polynomial Blending

The third blending function implemented corresponds to a piecewise polynomial function. The idea is to define the curvature of $f'(x)$ on each piece of the blending region. Notice that, because

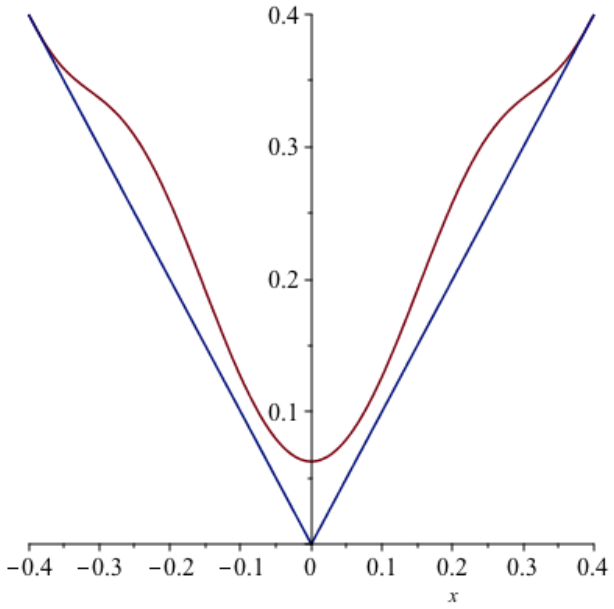

Figure 5: Example of function $f(x)$ which can generate non-convex shapes on blending regions.

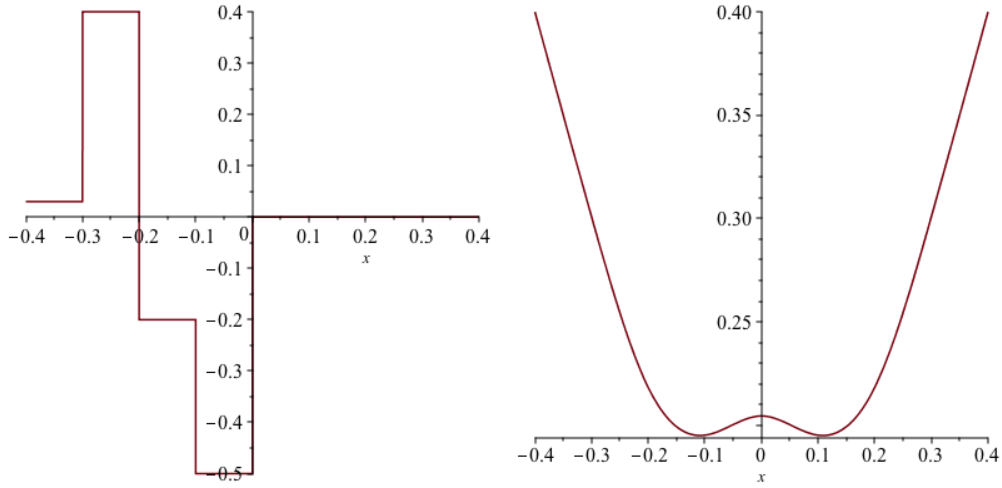

Figure 6: Example of curvatures chosen and corresponding result.

we are working with symmetric functions, we can define these curvatures only on the left side, for example, as shown in Figure 6 and then reflect it (with respect to y axis) on positive x values.

Then, we work in a similar way as before. First we integrate the curvature function $w(x)$:

$$r(x) = \int_{-2/\rho}^{x} w(z)dz$$

Then, we integrate again to obtain the curve $h(x)$:

$$h(x) = \int_{-2/\rho}^{x} r(z)dz$$

After this, we normalize to obtain values from -1 to 1. See also Figure 7.

$$g(x) = \frac{h(x)}{h(0)} - 1$$

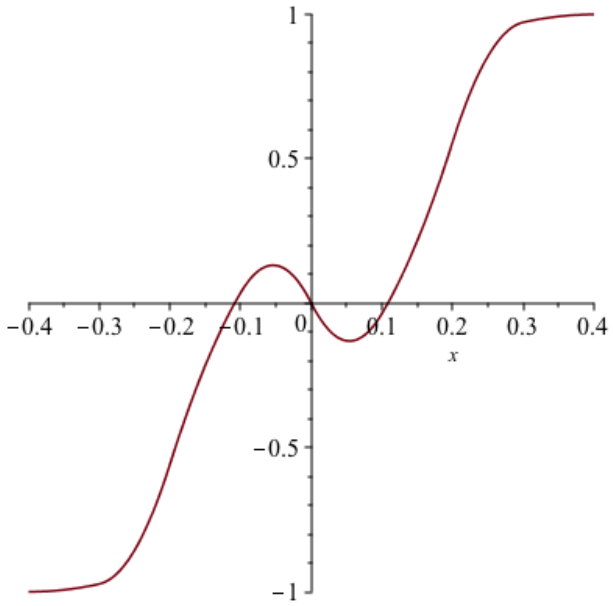

Figure 7: Function $g(x)$ representing slope of the final piecewise function.

Finally, we integrate one last time and normalize the result obtaining:

$$q(x) = \int_{-2/\rho}^{x} g(z)dz$$
$$f(x) = q(x) + 2/\rho - q(2/\rho)$$

The result can be seen in Figure 8. For more information, please take a look at script `KS_piecewise.mw` (which presents the recursive definition of this blending function used in the implementation) and `KS_piecewise_example.mw` (which presents an example with 4 pieces and the functions as described here).

5 Approximating the minimum of more than two numbers

In the case of more than 2 values, it becomes more difficult to apply the approximation of the absolute function directly as presented before. However, we know the following:

$$\min(y_1, y_2, y_3, \ldots, y_n) = \begin{cases} \min(\min(y_1, y_2, \ldots, y_{n-1}), y_n) & n > 2 \\ \min(y_1, y_2) & n = 2 \end{cases}$$

We need to be cautious here because the smooth approximation of min will not be symmetric in terms of the order of the arguments. For example, $\min(1, 2, 3)$ may be different from $\min(3, 2, 1)$,

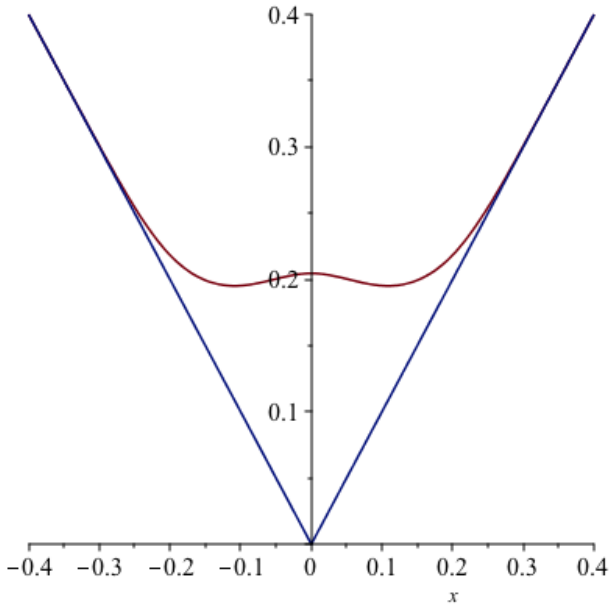

Figure 8: Example of a piecewise function $f(x)$, approximating $|x|$ smoothly.

depending on the approximation. That said, we can use the symmetric version shown below:

$$\min(y_1, y_2, y_3, \ldots, y_n) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \min(y_i, \min(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)) & n > 2 \\ \min(y_1, y_2) & n = 2 \end{cases}$$

Notice that we can implement this function in a recursive way. Now, we can simply use one of the 3 blending functions described above for each min call between two numbers. Also, notice that the approximations are propagated and, at the last level calls, you will not have the min operation between the original numbers. To avoid having high accumulation of these *errors*, at each recursive call the parameter related to blending region $s = 1/\rho$ is divided by a scalar $\alpha > 1$. So, if $\text{smin}_\rho()$ is our smooth minimum function with parameter ρ, we have:

$$\text{smin}_\rho(y_1, y_2, y_3, \ldots, y_n) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \text{smin}_\rho(y_i, \text{smin}_{\rho*\alpha}(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)) & n > 2 \\ \text{smin}_\rho(y_1, y_2) & n = 2 \end{cases}$$

References

[1] J. Panetta, A. Rahimian, and D. Zorin. Worst-case stress relief for microstructures. *ACM Transactions on Graphics*, 36(4), 2017.