Blending functions based on trigonometric and polynomial approximations of the Fabius function

Hans Olofsen
UiT The Arctic University of Norway in Narvik
hans.olofsen@uit.no

Abstract

Most simple blending functions are polynomials, while more advanced blending functions are, for example, rational or expo-rational fractions. The Fabius function has the required properties of a blending function, but is a nowhere analytic function and cannot be calculated exactly everywhere on the required domain. We present a new set of trigonometric and polynomial blending functions with the shape and other properties similar to the Fabius function. They consist of combinations of trigonometric polynomials and piecewise polynomials. The main advanced of these are that they are easy to implement, have low processing costs and have simple derivatives. This makes them very suitable for the calculation of splines. Due to the self-differential property of the Fabius function, scaled versions of these functions can even be used to approximate their own derivatives.

1 Introduction

In computer graphics, interpolation and approximation of points, curves and surfaces have always been an important topic. Many of the techniques used involve blending. A blending function is used to blend data or functions, and smoothness of the result increases with the order of the blending function.

For decades, trigonometric polynomials have been used for approximation. This has especially been the case for approximating periodic functions using Fourier series. Yet there has been little focus on using them use as blending functions, where the domain is $[0,1]$. This might be due to the fact that using a limited number of coefficients from the Fourier series in trigonometric polynomials does not result into a valid blending function where the endpoints have exact values of 0 and 1.

The Fabius function has the properties of blending functions, is C^∞-smooth and is therefore an interesting candidate for the use as a blending function. Unfortunately it is nowhere analytic and cannot be calculated exactly on the fully required domain. Using trigonometric series, the Fabius function can be approximated while still adhering to the properties of blending functions. The self-differentiability of the Fabius function can be used to find derivatives of the approximate functions, but also to find better approximate functions.

We will now first look at blending functions and the Fabius function, before we start looking at the approximate functions.

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

2 Preliminaries

Blending functions

Blending functions, or B-functions, are used for blending two other functions, e.g. for the interpolation of curves and surfaces. How exactly this is done is beyond the scope of this paper. For most applications, a (point) symmetric blending function is required. For simplicity, whenever a blending function is called symmetric, it implies that the function is point symmetric about the center point $\left(\frac{1}{2},\frac{1}{2}\right)$. As a consequence, all odd-order derivatives are line symmetric about the vertical center line $t=\frac{1}{2}$, while all even-order derivatives are point symmetric over the middle point $\left(\frac{1}{2},\frac{1}{2}\right)$.

Definition 2.1. A symmetric B-function has the properties, described in [1]:

- $B(t)\colon\ [0,1]\to[0,1]$ (P1)
- Has fixed endpoints: $B(0)=0$ and $B(1)=1$ (P2)
- Is continuous and monotone: $B'(t)\geq 0$ (P3)
- Is (point) symmetric: $B(1-t)=1-B(t)$ (P4)
- Has a Hermite order $\mathsf{S}\geq 0: B^{(j)}(0)=B^{(j)}(1)=0\,, j=1,\ldots,\mathsf{S}$ (P5)

Besides blending the values of the two functions, blending the derivatives of the two function is important for smooth results. The Hermite order determines up to which order derivatives the blending is interpolated. The simplest function $B(t)=t$ can be used to give a continuous result, but does not provide any smoothness at the border: $\mathsf{S}=0$.

The Fabius function

The Fabius function[2] is an infinitely differentiable, but nowhere analytic function that is self-differential:

$$\mathfrak{F}'(x)=2\,\mathfrak{F}(2x)\ ,\ x\in[0,\infty) \tag{2}$$

Figure 1 shows the Fabius function and its first 3 derivatives.

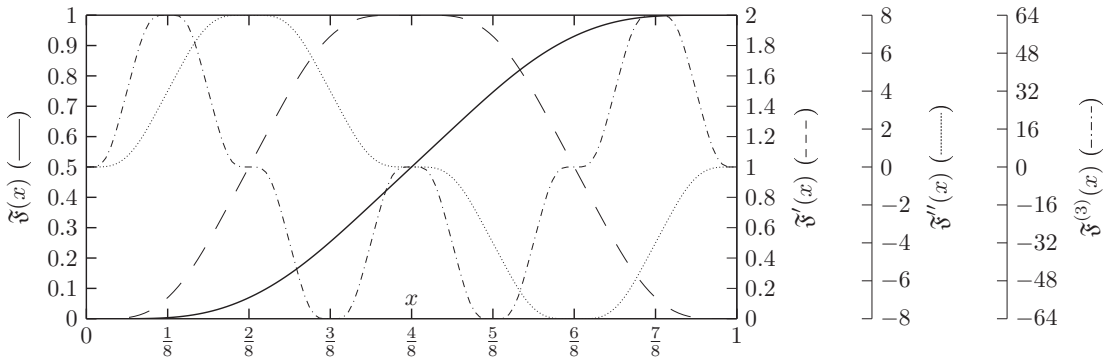


Figure 1: The Fabius function (——) and its first 3 derivatives (– – , ····· and – · –). Note the ranges of the four different y-axes.

On the domain $x\in[0,1]$, the Fabius function is point symmetric about $\left(\frac{1}{2},\frac{1}{2}\right)$. This makes it possible to limit the domain for evaluation of the Fabius to $x\in[0,1]$, by using the

"unextended definition", which is the piecewise definition

$$\mathfrak{F}(x) = \begin{cases} \int_0^{2x} \mathfrak{F}(\tau)\ \mathrm{d}\tau\ , & \text{if } x \in \left[0, \tfrac{1}{2}\right]\ , \qquad (3a) \\ \int_{2x-1}^{1} \mathfrak{F}(\tau)\ \mathrm{d}\tau\ +\ 2x{-}1\ , & \text{if } x \in \left(\tfrac{1}{2}, 1\right]\ . \qquad (3b) \end{cases}$$

In the domain $[0,1]$, the Fabius function has the basic properties for symmetric blending functions (P1) – (P5) and therefore it can be used as a blending function. It is infinitely smooth C^∞ and when used for blending, the resulting curve or surface will be C^∞-smooth as well. However, it cannot be calculated exactly (except for positive dyadic rational numbers, numbers of the form $\frac{m}{2^n}$, [3]) and therefore needs to be approximated.

3 Approximating the Fabius function

In order to find approximations $\mathcal{F}$ of the Fabius function, we have several options. Since the purpose of these approximate functions is the use as blending functions, they should adhere to the basic properties for symmetric blending functions (P1) – (P5). We will therefore use t as the argument of these functions. We can find function with a shape that is similar to the shape of the Fabius function. We can then also adjust the shape of the function such that its derivatives are similar to the derivatives of the Fabius function. We will show how this can be done with trigonometric polynomials.

We will also show how an approximation $\mathcal{F}(t)$ can be transformed into a better approximation $\widehat{\mathcal{F}}(t)$, and this process can be repeated in order to get better approximations, though at a computational cost.

Approximating with a sum of weighted cosines

A sum of weighted cosines can approximate the Fabius function on the domain $[0,2]$, similar to [4, eq.(30)], by the following trigonometric polynomials:

$$\mathfrak{F}(t) \approx \mathcal{F}_{\mathsf{M}}(t) = \frac{1}{2} - \sum_{m=1}^{\mathsf{M}} c_m \cos(\pi m t) \qquad (4)$$

Due to point symmetry (P4), it follows that $c_m = 0$ for m is even. We decide that M has only values that are rounded up to the nearest even number, and we do so consistently such that we can later use the value of M to calculate the Hermite order. M is then twice the number of non-zero c_m coefficients which is equal to the number of constraints.

From (P2) (endpoints) it follows that $\sum c_m = \frac{1}{2}$ which is the first constraint. As a consequence, the most simple trigonometric approximation, with one non-zero c_m is:

$$\mathcal{F}_{\mathsf{M}=2}(t) = \frac{1}{2} - \frac{1}{2}\cos(\pi t) \qquad (5)$$

Increasing the Hermite order

The Hermite order (P5) of (5) is then $\mathsf{S} = 1$. With $\mathsf{S} = \Delta\mathsf{S} - 1$, the Hermite order can be increased by $\Delta\mathsf{S}$ in steps of 2 by solving equations such that one additional derivative at

the endpoints vanishes. This results in a larger number of non-zero values for c_m. The following sets of equations need to be solved for $\Delta\mathsf{S}=2$ and $\Delta\mathsf{S}=4$:

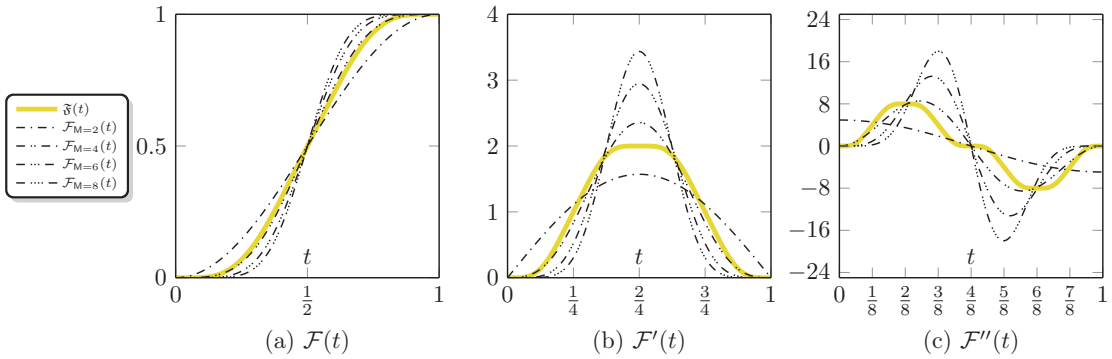
$$(\Delta\mathsf{S}=2)\quad c_{1,3}:\begin{cases} c_1+\quad c_3=\frac{1}{2} \\ 1^2c_1+3^2c_3=0 \end{cases};\tag{6}$$

$$(\Delta\mathsf{S}=4)\quad c_{1,3,5}:\begin{cases} c_1+\quad c_3+\quad c_5=\frac{1}{2} \\ 1^2c_1+3^2c_3+5^2c_5=0 \\ 1^4c_1+3^4c_3+5^4c_5=0 \end{cases}.\tag{7}$$

For $\Delta\mathsf{S}=2$, the solutions are $c_1=\frac{9}{16}$ and $c_3=-\frac{1}{16}$. We then get the following approximate function

$$\mathcal{F}_{\mathsf{M}=4}(t)=\frac{1}{2}-\left(\frac{9}{16}\cos(\pi t)+-\frac{1}{16}\cos(3\pi t)\right).\tag{8}$$

In Figure 2 we see that both $\Delta\mathsf{S}=2$ and $\Delta\mathsf{S}=4$ clearly give a better approximation, but higher values cause a convergence to a step function (e.g. Heaviside: $\mathcal{H}(t-\frac{1}{2})$) rather than to the Fabius function.



(a) $\mathcal{F}(t)$ (b) $\mathcal{F}'(t)$ (c) $\mathcal{F}''(t)$

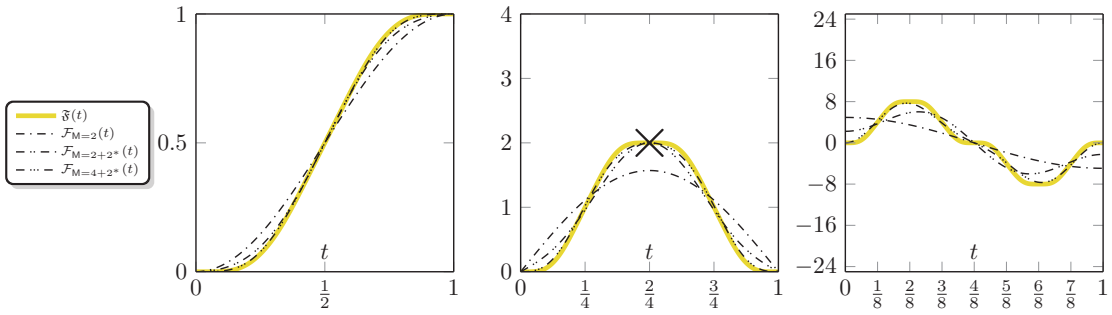
Figure 2: Increasing hermite order S by increasing the number of c_m with constraints. A higher $\Delta\mathsf{S}$ does not necessarily make a better approximation.

Other constraints

Increasing the Hermite order might be desired, but this alone does not improve the approximation. This can be solved by adding constraints for the derivatives at certain (dyadic rational) positions. These (and possibly other constraints) increase M, and this will be written as: $\mathsf{M}=\Delta\mathsf{S}+\mathsf{M}^*$, where M^* does not increase the Hermite order.
In the center of Figure 3 we can see that the second derivate also vanishes, but the third does not. This could be improved by adding more constraints, but requires even more values for c_m which requires more calculations. For $t\in\left[0,\frac{1}{2}\right]$, we see that $\mathfrak{F}''(t)$ is symmetric about $t=\frac{1}{4}$. However, the approximations with (4) are not. This is solved by using the antiderivatives as explained below.

Approximating by using the antiderivatives

So far we have focused on values of the Fabius function and its derivatives, which is, essentially, interpolating those. However, we can also focus on the main property of the



(a) $\mathcal{F}(t)$ (b) $\mathcal{F}'(t)$ (c) $\mathcal{F}''(t)$

Figure 3: A better approximation can be achieved by finding values for c_m such that $\mathcal{F}'(\frac{1}{2}) = \sum c_m \pi (-1)^{\frac{m-1}{2}}$ $m = 2$ (marked with $\times$).

Fabius function that defines its shape, namely the self-differential property. Suppose $\mathcal{F}(t)$ is an approximation to the Fabius function $\mathfrak{F}(t)$, with error $\mathcal{E}(t)$:

$$\mathcal{F}(t) = \mathfrak{F}(t) + \mathcal{E}(t)\,, \tag{9}$$

then, similar to (3), we define $\widehat{\mathcal{F}}(t)$ as a "new" approximation with error $\widehat{\mathcal{E}}(t)$:

$$\widehat{\mathcal{F}}(t) = \begin{cases} \displaystyle\int_0^{2t} \mathcal{F}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \mathcal{F}(\tau)\,\mathrm{d}\tau + 2t - 1\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{10a}$$

$$= \mathfrak{F}(t) + \begin{cases} \displaystyle\int_0^{2t} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{10b}$$

$$= \mathfrak{F}(t) + \widehat{\mathcal{E}}(t)\,, \tag{10c}$$

where $\widehat{\mathcal{E}}(t) = -\widehat{\mathcal{E}}(1-t)$ is the error of the new approximation:

$$\widehat{\mathcal{E}}(t) = \begin{cases} \displaystyle\int_0^{2t} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[\frac{1}{2}, 1\right]. \end{cases} \tag{11}$$

In Figure 4 and 5 the error $\mathcal{E}(t)$, its definite integral $\int_0^t \mathcal{E}(\tau)\,\mathrm{d}\tau$ and the new error $\widehat{\mathcal{E}}(t)$ are shown. We can see that the new approximation is better, since both the average and the maximum of the error become smaller. This is true because the width of the domain $[0,1]$

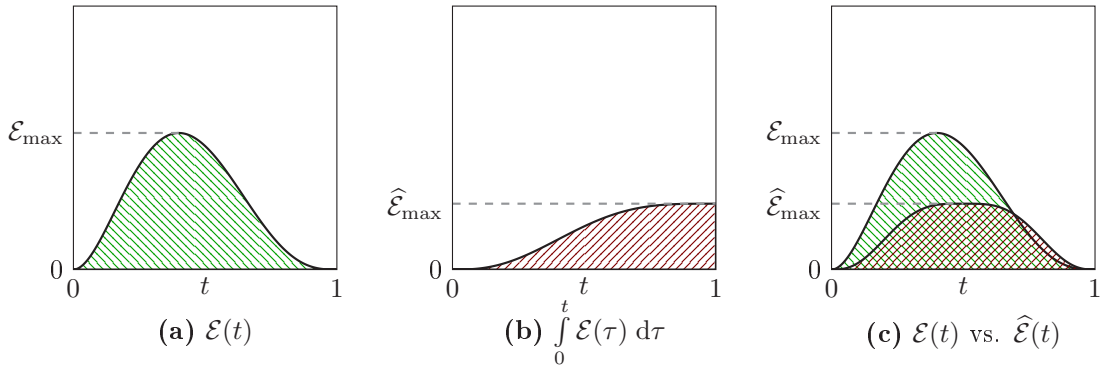


Figure 4: An example where the error of an approximation to the Fabius function doesn't change sign on the domain $[0,1]$. For $\mathsf{P} > 1$ it shows that $\widehat{\mathcal{E}}_{max}$ is about twice ass small as $\mathcal{E}_{\max}$.

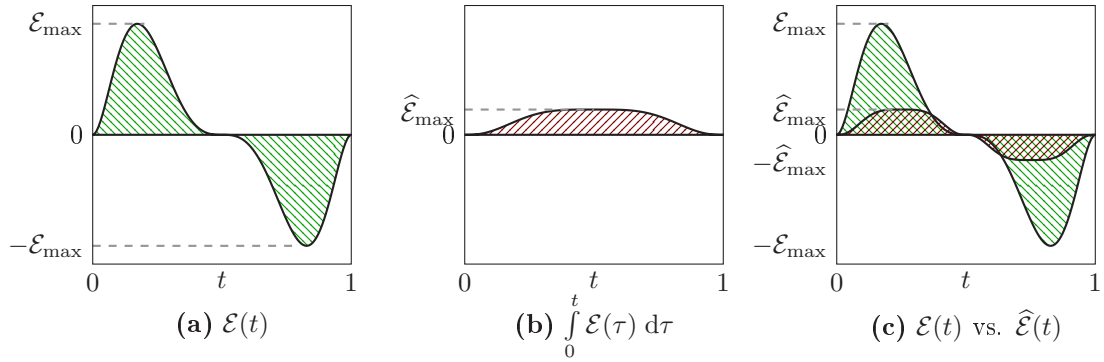


Figure 5: An example where the error of an approximation to the Fabius function changes sign at $t = \frac{1}{2}$. This is the case for all approximations discussed in this paper. For $\mathsf{P} > 1$ it shows that $\widehat{\mathcal{E}}_{max}$ is about 4 times smaller than $\mathcal{E}_{\max}$.

is not more than 1. Summarized this is the following inequation:

$$\mathcal{E}_{\max} = \max_{0<t<1} \mathcal{E}(t) > \int_0^1 \mathcal{E}(\tau)\,\mathrm{d}\tau = \widehat{\mathcal{E}}(1) = \widehat{\mathcal{E}}_{max} \tag{12}$$

In order to get a better approximation, we can then replace $\mathcal{F}$ by $\widehat{\mathcal{F}}$ and repeat this process. This process can be repeated in order to get an even better approximation, where P is the number of times:

$$\mathcal{F}_{\mathsf{P}}(t) = \widehat{\mathcal{F}}_{\mathsf{P}-1}(t)\ ,\ \text{for}\ \mathsf{P} > 1\ . \tag{13}$$

It should be noted that (12) is not true when $\mathcal{E}(t)$ is constant. However if the process is repeated, it will be true as $\widehat{\mathcal{E}}(t)$ won't be constant.

The cost of each improvement is more complex functions due to piecewise definitions. It might seem that each time, the number of sub-functions should double, but fortunately this is not always the case as we will see later. Now that we have a way to make better approximations, we should find a way to start with a first approximation. It should be noted that, in order to find a valid blending function, it is not necessary to start with a valid blending function as long as the process is repeated until it results in a valid blending function. The only requirement are that $\int_0^1 \mathcal{F}(\tau)\,\mathrm{d}\tau = \frac{1}{2}$ and that the antiderivative is chosen such that $\widehat{\mathcal{F}}(0) = 0$ and $\widehat{\mathcal{F}}(t)$ is continuous. For example, it is possible to start

with

$$\mathcal{F}_{\mathsf{P}=0}(t) = \frac{1}{2}\,, \tag{14}$$

which clearly is not a valid blending function since it does not satisfy (P2), and, one could say in this context, has "Hermite order -1". However, after one improvement step, we get the following function which is a valid blending function:

$$\mathcal{F}_{\substack{\mathsf{P}=1\\ \mathsf{M}=0}}(t) = \begin{cases} t\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ 1 - t \;+\; 2t - 1\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} = t\,. \tag{15}$$

We see that the two sub-functions are that same, and therefore the number of sub-function doesn't double in this case.

In Figure 6 the approximations for $\mathsf{P} = 1, \ldots, 5$ are shown, as well as the maximum errors $\mathcal{E}_{\max}$ for $\mathsf{P} = 1, \ldots, 6$. After several iterations of using the proper antiderivative, the resulting piecewise polynomial shape approximates the Fabius function quite well. Figure 6 (b) shows the two sub-functions, separated by bullets, of the approximation

$$\mathcal{F}_{\mathsf{P}=2}(t) = \begin{cases} 2t^2\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ 1 - 2(1-t)^2\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{16}$$

They are are not the same, and therefore here the number of sub-function doubles. As a consequence, the function is no longer C^∞-smooth as summarized in (23).

$$\mathcal{F}_{\mathsf{P}=3}(t) = \begin{cases} \displaystyle\int_0^{2t} 2\tau^2 \,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{4}\right]; \\ \displaystyle\int_0^{1/2} 2\tau^2 \,\mathrm{d}\tau \;+\; \int_{1/2}^{2t} 1 - 2(1-\tau)^2 \,\mathrm{d}\tau\,, & \text{if } t \in \left[\frac{1}{4}, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1/2} 2\tau^2 \,\mathrm{d}\tau \;+\; \int_{1/2}^{1} 1 - 2(1-\tau)^2 \,\mathrm{d}\tau \;+\; 2t - 1\,, & \text{if } t \in \left[\frac{1}{2}, \frac{3}{4}\right]; \\ \displaystyle\int_{2t-1}^{1} 1 - 2(1-\tau)^2 \,\mathrm{d}\tau \;+\; 2t - 1\,, & \text{if } t \in \left[\frac{3}{4}, 1\right], \end{cases} \tag{17a}$$

but, again, the to sub-functions in the middle are the same:

$$\cdots = \begin{cases} \frac{16}{3} t^3\,, & \text{if } t \in \left[0, \frac{1}{4}\right]; \\ \frac{16}{3}\left(\frac{1}{2} - t\right)^3 - \frac{1}{2} + 2t\,, & \text{if } t \in \left[\frac{1}{4}, \frac{3}{4}\right]; \\ 1 - \frac{16}{3}(1-t)^3\,, & \text{if } t \in \left[\frac{3}{4}, 1\right], \end{cases} \tag{17b}$$

The sub-functions are slightly rewritten such that they show the symmetry. Figure 6 (c) shows these three sub-functions.

Due to point symmetry, it is clear that the sub-functions in the middle are the same for odd values of P, which is the case for odd polynomials. The number of pieces for $\mathsf{P} = 0, 1, 2, \ldots$ are $1, 1, 2, 3, 6, 11, 22, 43, \ldots$ which is sequence A005578 in the *On-Line Encyclopedia of Integer Sequences*[5]. One can verify that the number of pieces converges to $\frac{2^{\mathsf{P}}}{3}$.

The combination of these

A better approximation can be achieved by starting first with (4) and then use the antiderivatives. This way one gets a combination, namely the sum of a polynomial and a trigonometric polynomial. The result is a non-periodic function, however we are only interested in the domain $[0, 1]$.

The same way as in (15), since $\cos(\pi m t)$ is symmetric in $t = 1$, we get two sub-functions that are the same. As a consequence the functions $\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}}}(t)$ simplify easily:

$$\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}}}(t) = \mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}=0}}(t) - \begin{cases} \displaystyle\int_0^{2t} \sum_{m=1}^{\mathsf{M}} c_m \cos(\pi m \tau)\,\mathrm{d}\tau, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \sum_{m=1}^{\mathsf{M}} c_m \cos(\pi m \tau)\,\mathrm{d}\tau, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{18}$$

$$= t - \sum_{m=1}^{\mathsf{M}} \frac{c_m}{\pi m} \sin(2\pi m t) \tag{19}$$

Figure 7 shows $\mathsf{P} = 0, \ldots, 3$ for increased Hermite order $\mathsf{M} = 2, 4$. Figure 8 shows $\mathsf{P} = 0, \ldots, 3$ with the additional $\mathcal{F}'(\frac{1}{2}) = 2$ constraint for $\mathsf{M} = 2 + 2^*, 4 + 2^*$.

4 Results

We now have several approximations of the Fabius function available that can be used for blending. Table 1 shows the maximum difference between these functions and the Fabius function. One can see that every increment of P causes an up to about 4 times lower maximum error. For blending in practice, we do not have to minimize this difference, and we can choose approximation that are good enough. For $\mathsf{P} > 0$ this maximum difference is at $t = \frac{1}{4}$ where $\mathfrak{F}\left(\frac{1}{4}\right) = \frac{5}{72} \approx 6.9 \times 10^{-2}$ as evaluated in [3]. Take notice of $\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}=0}}(t) = t$ which is the simplest B-function.

The Hermite order of the functions presented here is:

$$\mathsf{S} = \mathsf{P} + \Delta\mathsf{S} - 1. \tag{20}$$

When a high Hermite order is not required, and speed and simplicity are priorities, some of the simplest are the C^∞-smooth single-piece functions

$$\mathcal{F}_{\substack{\mathsf{P}=0\\\mathsf{M}=2}}(t) = \frac{1}{2} - \frac{1}{2}\cos(\pi t) \tag{5 revisited}$$

and

$$\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}=2}}(t) = t - \frac{\sin(2\pi t)}{2\pi} \tag{21}$$

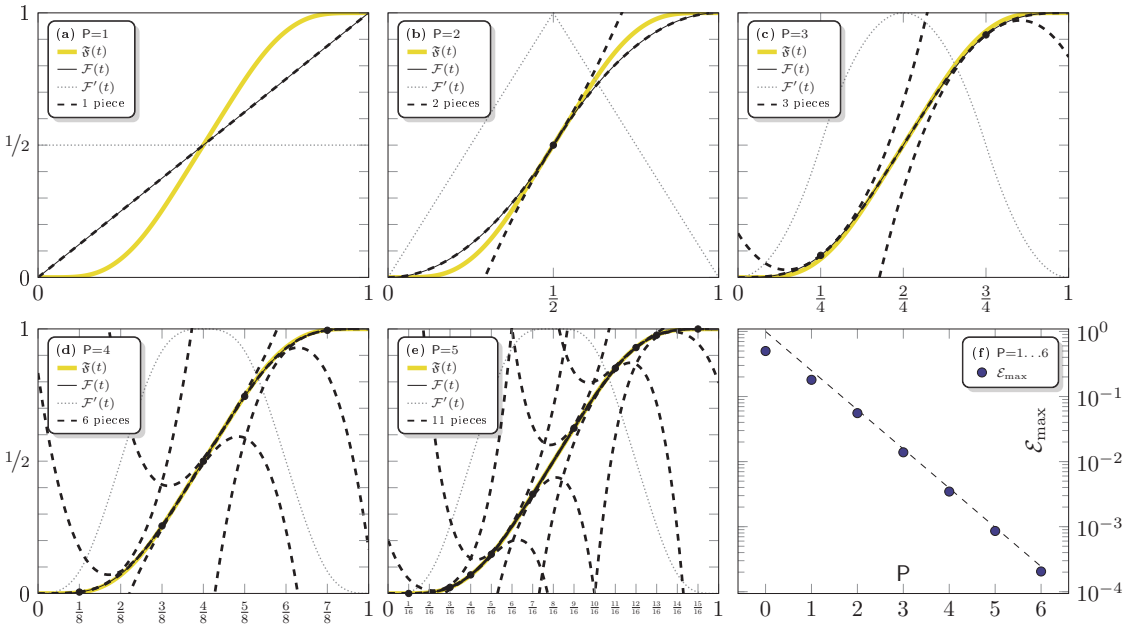


Figure 6: (a)-(e): Approximations for $\mathsf{P} = 1, \ldots, 5$ (f): the maximum errors $\mathcal{E}_{\max}$ for $\mathsf{P} = 1, \ldots, 6$.

In this figure and in Figure 7 and 8, the sub-functions (- - -) of the piecewise-defined functions are plotted for the whole domain; the first derivatives (·····) are scaled by a factor of 2.

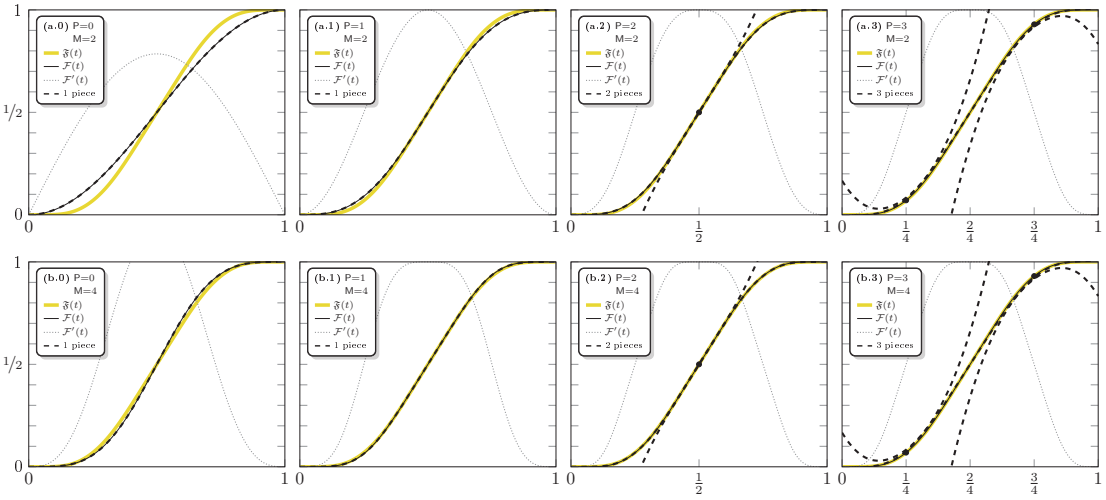


Figure 7: Approximations for $\mathsf{P} = 0, \ldots, 3$; $\mathsf{M} = 2, 4$

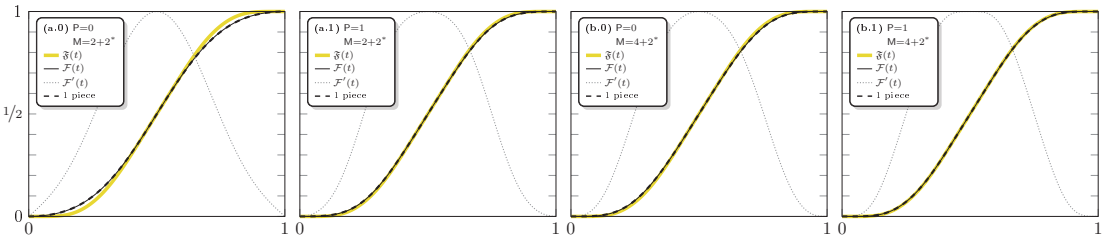


Figure 8: Approximations for $\mathsf{P} = 0, 1$; $\mathsf{M} = 2+2^*, 4+2^*$

P	M = 0	M = 2	M = 4	M = 2 + 2*	M = 4 + 2*
0	(not a valid B-function)	$7.8{\cdot}10^{-2}$	$2.6{\cdot}10^{-2}$	$2.9{\cdot}10^{-2}$	$9.1{\cdot}10^{-3}$
1	$1.8{\cdot}10^{-1}$	$2.1{\cdot}10^{-2}$	$5.1{\cdot}10^{-3}$	$6.9{\cdot}10^{-3}$	$4.3{\cdot}10^{-3}$
2	$5.6{\cdot}10^{-2}$	$4.9{\cdot}10^{-3}$	$7.4{\cdot}10^{-4}$	$1.8{\cdot}10^{-3}$	$4.8{\cdot}10^{-4}$
3	$1.4{\cdot}10^{-2}$	$1.2{\cdot}10^{-3}$	$1.9{\cdot}10^{-4}$	$4.4{\cdot}10^{-4}$	$1.1{\cdot}10^{-4}$
4	$3.5{\cdot}10^{-3}$	(not calculated)	(not calculated)	(not calculated)	(not calculated)
5	$8.6{\cdot}10^{-4}$	(not calculated)	(not calculated)	(not calculated)	(not calculated)
6	$2.0{\cdot}10^{-4}$	(not calculated)	(not calculated)	(not calculated)	(not calculated)

Table 1: Maximum absolute difference $\mathcal{E}_{\max}$ between a few approximated Fabius functions and the Fabius function. In the top row: $\mathsf{M} = \Delta\mathsf{S} + \mathsf{M}^*$

with $\mathsf{S} = 1$ and $\mathsf{S} = 2$, as well as the piecewise-defined polynomial function

$$\mathcal{F}_{\substack{\mathsf{P}=2\\ \mathsf{M}=0}}(t) = \begin{cases} 2x^2, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ 1 - 2(1-x)^2, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{22}$$

which is only C^1 smooth, because its first derivative is not differentiable at $t = \frac{1}{2}$. Other approximations can be chosen depending on requirements for the Hermite order, smoothness class, as well as implementation specific calculation optimizations such as parallelization.

Smoothness The smoothness class of the functions depends on P, since for $\mathsf{P} > 1$ piecewise defined functions are required and is:

$$\mathsf{C} = \begin{cases} \mathsf{C}^\infty, & \text{if } \mathsf{P} \leq 1; \\ \mathsf{C}^{\mathsf{P}-1}, & \text{if } \mathsf{P} > 1. \end{cases} \tag{23}$$

The sub-functions that define the pieces of the piecewise functions are always C^∞-smooth.

5 Applications

To illustrate how easy it is to implement all these functions, all figures so far in this paper have been calculated in TikZ, instead of using external tools or a precalculated data file.

While we haven't done any comparative benchmarking on these function, it is clear that the implementation is straight forward. Both functions (5) and (22) require only three basic arithmetic operations and one trigonometric function call/lookup that most platforms have implemented in hardware. No further interpolation is required. For functions with $\mathsf{P} > 1$ the speed will depend on the platform's branching mechanism.

An application: Fabius B-splines

The (approximations of) the Fabius function can be used in B-splines. In this case the linear translation and scaling function

$$w_{d,i}(t) = \begin{cases} \dfrac{t - t_i}{t_{i+d} - t_i}, & \text{if } t_i \leq t < t_{i+d} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

is replaced by $B \circ w_{d,i}(t)$ where the B-function B in this case is the Fabius function or one of the approximate functions $\mathcal{F}$.

Figure 9 shows the Cubic B-spline basis functions for normal polynomial B-splines, for Fabius B-splines where the Fabius function is used as the B-function B, as well as approximations (5) and (21). The curves made with these splines have a high smoothness, increased by the Hermite order S of the function $\mathcal{F}$, and even C^∞-smooth in case of Fabius B-splines.

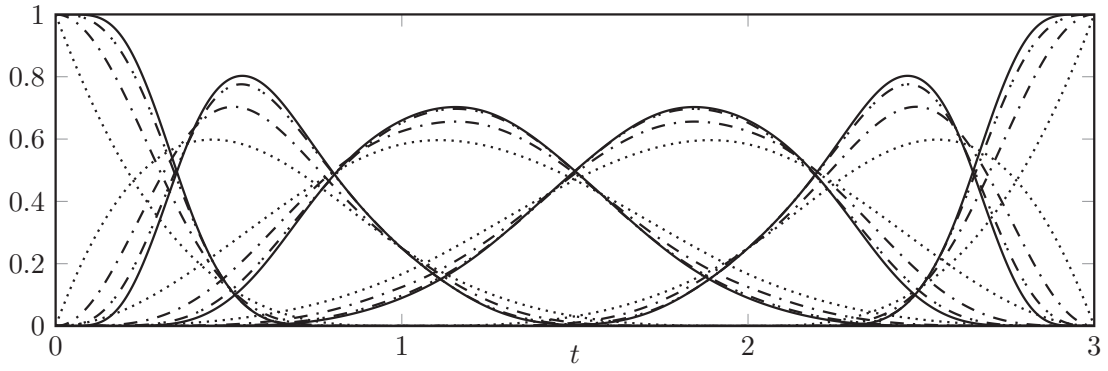


Figure 9: The Cubic B-spline basis functions for four different B-functions on knot vector $\vec{t} = \{0,0,0,0,1,2,3,3,3,3\}$.
····· Normal polynomial B-splines: $B(t) = t$
– · – B-splines using $B(t) = \mathcal{F}_{\substack{\mathsf{P}=0\\ \mathsf{M}=2}}(t) = \frac{1}{2} - \frac{1}{2}\cos(\pi t)$ from (5).
– - – B-splines using $B(t) = \mathcal{F}_{\substack{\mathsf{P}=1\\ \mathsf{M}=2}}(t) = t - \frac{\sin(2\pi t)}{2\pi}$ from (21).
—— Fabius B-splines: $B(t) = \mathfrak{F}(t)$.

Approximation of the derivatives

In most applications the derivatives on the blending function are needed. The smoothness of the approximate functions is determined by their differentiability. The derivative of a C^k-smooth function is C^{k-1}-smooth and as such the Hermite order of the derivative of a blending function is also lower then the Hermite order of that blending function. The derivative of an approximate Fabius function is in general easy to calculate, as it consists of only polynomials and trigonometric polynomials. But it is also a (piecewise) function where the sub-functions are scaled and mirrored versions of the approximate function with $\mathsf{P}-1$. As an example, compare the $\wedge$-shape of the derivative $\mathcal{F}'_{\mathsf{P}=2}(t)$ in Figure 6 (b) to the /-shape of $\mathcal{F}_{\mathsf{P}=1}(t)$ in Figure 6 (a). This means that the derivatives are worse approximations than the approximate functions.

However, since the functions are only approximations, the true derivatives of the functions are not so interesting. To approximate the derivatives, we can use the self-differentiability of the Fabius function. We define the *Fabius-derivative* $\widetilde{\mathfrak{D}}$ of a function $f(x)$ on the domain $[0,1]$ as:

$$\widetilde{\mathfrak{D}}\, f(x) = \begin{cases} 2f(2x)\,, & \text{if } x \in \left[0, \dfrac{1}{2}\right]; \\ 2f(2-2x)\,, & \text{if } x \in \left[\dfrac{1}{2}, 1\right]. \end{cases} \tag{25}$$

If f is C^a-smooth and has Hermite order $\mathsf{S} = b$ (at $x = 1$), then $\overset{\smile}{\mathfrak{D}} f(x)$ is $\mathsf{C}^{\min(a,b)}$-smooth and has Hermite order $\mathsf{S} = b$. It follows that higher order Fabius-derivatives have the same smoothness and Hermite order as the first order Fabius-derivative.

If one can accept that derivatives are also approximated, one can say that if a blending function $\mathcal{F}(t)$ approximates the Fabius function, its Fabius-derivative approximates the derivative of the Fabius function and $\mathcal{F}(t)$ is approximately C^∞-smooth and has approximately Hermite order $\mathsf{S} = \infty$. For the presented approximate functions this means that

$$\overset{\smile}{\mathfrak{D}}\, \mathcal{F}_{\mathsf{P}}(t) = \mathcal{F}'_{\mathsf{P}+1}(t). \tag{26}$$

Mathematically seen this can be considered as cheating or simply incorrect. However for practical applications this trick can be acceptable and quite useful.

6 Conclusion

We have presented new types of trigonometric and polynomial blending functions with the shape and properties similar to the Fabius function. We did this by showing how to find trigonometric polynomial blending functions with shapes that approximate the Fabius function, including some of their derivatives.

Then, using the self-differentiability of the Fabius function, we have shown how an approximation of the Fabius function can be improved iteratively, while retaining the blending function properties. This resulted in a set of blending functions that can be evaluated fast, because they can be calculated directly without needing further interpolation.

The blending functions have been successfully used in extended B-splines, resulting in smoother splines than normal B-splines.

References

[1] Arne Lakså. Beta-function B-splines and subdivision schemes, a preliminary study. In Ivan Lirkov and Svetozar Margenov, editors, *Large-Scale Scientific Computing*, pages 592–599, Cham, 2018. Springer International Publishing.

[2] J. Fabius. A probabilistic example of a nowhere analytic C^∞-function. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 5:173–174, 1966.

[3] J. K. Haugland. Evaluating the Fabius function. *ArXiv e-prints*, September 2016.

[4] J. Arias de Reyna Martínez. Definition and study of an infinitely differentiable function with compact support. *Rev. Real Acad. Cienc. Exact. Fís. Natur. Madrid*, 76(1):21–38, 1982.

[5] OEIS Foundation Inc. The On-Line encyclopedia of integer sequences. A005578, 2019. [Online; accessed 2019-08-01].