A new deterministic heuristic knots placement for B-Spline approximation

D. Michel, A. Zidna

Version of Record: https://www.sciencedirect.com/science/article/pii/S0378475420302482
Manuscript_80014693ea36688fc086e3794c2d8a22

A new deterministic heuristic knots placement for B-Spline approximation

D. Michel[a], A. Zidna[a,*]

[a] *LGIPM, University of Lorraine, Metz, France, Ile du Saulcy, F-57045 METZ.*

Abstract

In this paper, we propose an adaptive knot placement algorithm for B-Spline curve approximation to dense and noisy 2D data points. The proposed algorithm is based on a heuristic rule for knot placement. It consists in constructing a distribution knot function by blending geometric criteria such as discrete derivatives, discrete angular variations and curvature. It has been successfully compared to three well known methods for approximating various noisy functions and sets of data in handwriting context.

Keywords: Dense and noisy data, B-Spline approximation, Knot placement, Discrete derivatives, Discrete angular variation, Curvature, Handwriting compression.

1. Introduction

Curve fitting is a fundamental problem in many fields, such as computer graphics, image processing, shape modeling and data mining. It has become a fundamental tool in Reverse Engineering, where very dense data point sets are acquired from physical objects in order to refine a digital model of such objects. Usually, the underlying function of data is difficult to approximate through a simple function. The B-Spline approximating functions are widely used as they are very convenient and can represent a large variety of shapes. This is due to their powerful mathematical properties such as local modification, projective invariance and convex hull property. The knots placement of the approximation B-spline curve plays an effective role in order to recover the underlying curve of the data points. When the given data points present considerable curvature variation of the underlying curve, the reconstructed approximation curve may be significantly different from the original curve if the knots are not distributed according to the varying of the curvature. The approximation of functions by B-Splines is greatly improved if the knots are treated as free variables. In many applications, curve approximation methods are iterative process. In the conventional approach to this problem, there are two ways for knots placement. One starts with the minimum (or the maximum) of knots and increase (or reduce) the number of knots to satisfy the error bound specified between the original curve and the approximation curve [6, 10, 11]. Unfortunately, this process fails to automatically generate a good knot vector and is time consuming if the initial knots are not well chosen.

For some years now, the trend in this topic has been to develop stochastic methods (genetic algorithms, swarm particle algorithms, etc.). They have been proved efficient in regard of the accuracy of the solution. Goldenthal et al.[9] consider to solve the problem as a minimization of a cost function that depends on the knots. They propose to use a genetic algorithm. In the same way, the authors propose a real coded genetic algorithm for data fitting with a B-Spline [13]. In [12], the authors propose to use artificial immune system (AIS) strategy to determine the locations and counts of interior knots of B-spline curves that fit a given data. Their benefits of using AIS is to accelerate the convergence to the objective function. In [1], in order to automatically compute an appropriate location of knots, the authors propose to use the particle swarm optimization (PSO) paradigm. This scheme yields very accurate results, even for curves with singularities and/or cusps. In [18, 19], a Multi-Objective-Genetic Algorithm has been developed, to optimize both the

*Corresponding author.
Email address: ahmed.zidna@univ-lorraine.fr (A. Zidna)

number of knots and its optimal placement for cubic or bicubic interpolating B-Splines. This technique is able to avoid the large number of local minima existing in the problem of random knots placement. But all of these methods suffer a huge drawback : the CPU-time needed to achieve good results.

More recently, In [17], the Support Vector Machines (SVMs) method is used to determine suitable knot vectors for B-spline curve approximation. The input for SVMs is a sequential points cloud, and the output from SVMs is the points set with signed scores that measures the quality of each location. Here the score is based on geometric and differential geometric features of points. But there are many other related works. For instance in [15], the authors propose an optimal method to fix the location and the number of knots in curve fitting with B-Splines by solving a sparse optimization problem. The objective is to minimize the total jump in the third derivatives on the knots. In [16], the authors propose a framework for computing the number of knots and the appropriate positions for spline fitting based on the active knot paradigm. Starting with an initial dense knot vector, they remove the inactive knots and the remaining ones become the candidate knots. In practice, They also adopt strategies to automatically adjust the number and location of candidate knots. The proposed method significantly improves the one described in [15]. However, it is very sensitive to noise due to the sensitivity of the third derivative. In [14], a heuristic algorithm is presented to approximate scattered data with a B-spline surface. An heuristic criterion for optimal knot positions in the fitting problem is formulated as an optimization problem according to the geometric feature distribution of the input data. Then, the coordinate descent algorithm is used for the optimal knot computation. In [2], the authors introduce the concept of dominant points which leads to an efficient algorithm. This approach is more detailed in the following as we used it as a method to compare our algorithm with. In this paper, we propose a deterministic heuristic method to set the knots. It is based on the blending of different geometric features of the input data such as discontinuities of successive derivatives, curvature and angular variations of derivatives. It has been successfully compared to a classical gradient method, to a PSO algorithm and to Park method [2]. This paper is organized as follows. In section 2, we introduce the B-spline curve approximation. Then, in section 3, the new method is fully described. Section 4 deals with experiments, two benchmarks of data clouds are set. The first one consists in a series of six noisy non parametric functions. The second one mainly consists in naturally noisy handwritten words. Computational time and error measuring are discussed. Finally, in section 5, the conclusion is given and further work is discussed.

2. B-spline approximation

Let us denote the cloud of data P as follows : $P = \{(t_0, \boldsymbol{P_0}), \ldots, (t_N, \boldsymbol{P_N})\}$ where the t_i are the time coordinates of the points and the $\boldsymbol{P_i}$ are the space coordinates of the points. That is, $\{t_i\}_{i=0\ldots N}$ is an increasing sequence of real numbers and the $\boldsymbol{P_i}$ belong to $\mathbb{R}^d$ $(2 \leq d)$. By an affine transform, the finite sequence $\{t_0, \ldots, t_N\}$ can always be replaced by a sequence which entirely lies in $[0, 1]$ with $t_0 = 0$ and $t_N = 1$. In the following, we assume it is the case. Furthermore, we assume that, by linear interpolation, it is always possible to calculate a "missing point" in between two consecutive points of P. The general problem discussed here is to design a B-Spline curve C [3, 4] which closely approximates a finite cloud of points P. These data points are supposed to lie on some unknown curve $t \mapsto L(t)$ and are sorted along the time axis. They were obtained from some physical measures, hence noisy in the general case. The obvious purpose of this endeavour is data compression since the amount of necessary points dramatically drops when passing from a discrete sampled curve to a B-Spline curve. A B-Spline curve C of order k is a piecewise polynomial of degree $k-1$ defined by a set of $m+1$ control points $\{\boldsymbol{c_i}\}_{i=0\ldots m}$ and by a set of knots $\{u_i\}_{i=0\ldots m+k}$. The greater the value of k, the smoother the curve C. The $\boldsymbol{c_i}$ are defined over the space and the u_i are real numbers distributed along the time axis. The problem addressed by this paper can be expressed as follows : Given m, k and a noisy discrete sampled curve P : calculate the knots then the control points of a B-Spline curve that minimizes the error (L2-norm) between the given data curve and the resulting B-Spline curve. Once the knots are set, the control points can straightforwardly be found using a well conditioned linear system based on L2-Inner product. The challenging part is to calculate the knots for this is a highly non linear problem.

2.1. B-Spline model

Usually, C is given by the following definition : Let m and k be two integer numbers such that $1 \leq k \leq m+1$. Let $U = \{u_0, u_1, u_2, \ldots, u_{m+k}\}$ be a finite sequence of real numbers such that : $u_i \leq u_{i+1}$ for all $i = 0 \ldots m+k-1$ and $u_i < u_{i+k}$ for all $i = 0 \ldots m$. Then $C : \mathbb{R} \to \mathbb{R}^d$ is defined as follows :

$$t \mapsto \boldsymbol{c(t)} = \sum_{i=0}^{m} \boldsymbol{c_i} N_i^k(t) \tag{2.1}$$

where the coefficients $\boldsymbol{c_i}$ are given in the d-dimensional euclidian space (in the following, we assume that $d = 2$) and the $t \mapsto N_i^k(t)$ are the basis of all the B-Spline functions over U. They are defined as follows:

$$N_i^1(t) = \begin{cases} 1, & \text{for} \quad u_i \leq t < u_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{2.2}$$

and for $2 \leq r \leq k$, we have

$$N_i^r(t) = \frac{t - u_i}{u_{i+r-1} - u_i} N_i^{r-1}(t) + \frac{u_{i+r} - t}{u_{i+r} - u_{i+1}} N_{i+1}^{r-1}(t). \tag{2.3}$$

Under these hypotheses, the $\boldsymbol{c_i}$ are called *control points*, the u_i *knots* and U a *knot vector*. The outer knots $u_0, \ldots, u_{k-1}$ and $u_{m+1}, \ldots, u_{m+k}$ are usually set such that $u_0 = \ldots = u_{k-1} = 0$ and $u_{m+1} = \ldots = u_{m+k} = 1$. This ensures that $\boldsymbol{c(0)} = \boldsymbol{c_0}$ and that $\boldsymbol{c(1)} = \boldsymbol{c_m}$, that is, endpoints of the curve C coincide with endpoints of the polygonal chain $\boldsymbol{c_0} \ldots \boldsymbol{c_m}$. Under these assumptions, the knot vector U is said *clamped*. In the following, all knot vectors comply with this property.

2.2. Fitness function

The input of the problem is P and the output is m, k, U and $\{\boldsymbol{c_i}\}_{i=0\ldots m}$ which ensure that C fits as best as possible the cloud P. In the following, we always assume that the cloud P is dense in regard to the set of control points of C, that is, $N \gg m$. Once m, k and U are set, the coefficients $\boldsymbol{c_i}$ are determined through solving linear systems of equations (least square method, quasi-interpolation techniques, etc.). The hardest part is how to define U. Since it is a solution of a highly non linear system of equations, U, that is, the inner knots $u_k, \ldots, u_m$ must be set otherwise. As we are dealing with an optimization problem, a convenient way to describe the whole process is to define a fitness function F. Let m, k and P be fixed, then $F : [0,1]^{m-k+1} \to \mathbb{R}^+$ is defined as follows :

$$(u_k, \ldots, u_m) \mapsto F(u_k, \ldots, u_m) = \frac{|L - C|^2}{|L|^2} \tag{2.4}$$

where $\frac{|L-C|^2}{|L|^2}$ is the squared relative error between the two curves $t \mapsto C(t)$ and $t \mapsto L(t)$. Let us detail how F is computed.

2.3. Computing the control points $\boldsymbol{c_0}, \ldots, \boldsymbol{c_m}$

First, by linearly interpolating $P = \{(t_0, \boldsymbol{P_0}), \ldots, (t_N, \boldsymbol{P_N})\}$, we deduce the continuous curve $t \mapsto L(t)$. This is made possible for P is supposed to be dense. Then, we consider the inner product in the vector space of continuous functions over $[0,1]$. That is, given two continuous functions $f : [0,1] \to \mathbb{R}$ and $g : [0,1] \to \mathbb{R}$, the inner product of f and g is $< f, g >= \int_0^1 f(t)g(t)\mathrm{d}t$. Then the control points $\boldsymbol{c_0}, \ldots, \boldsymbol{c_m}$ are calculated by projecting $t \mapsto L(t)$ on the vector space of B-Spline functions. Let $\boldsymbol{c(t)} = \sum_{i=0}^{m} \boldsymbol{c_i} N_i^k(t)$ be that projection. Then $\forall i = 0 \ldots m$, we have

$$< N_i^k, L >=< N_i^k, \boldsymbol{c} >=< N_i^k, \sum_{j=0}^{m} \boldsymbol{c_j} N_j^k >$$

Finally, we get :

$$< N_i^k, L >= \sum_{j=0}^{m} < N_i^k, N_j^k > \boldsymbol{c_j} \quad \forall i = 0 \dots m \tag{2.5}$$

which defines a very well conditioned $(m+1) \times (m+1)$ linear system of which the solutions are the $\boldsymbol{c_i}$. We solve it using Gauss-Jordan global pivot reduction. Strictly speaking, in the three equations above, $<,>$ refers to an integral but not always to the inner product since the left member $< N_i^k, L >$ is actually not a scalar. We write it this way for the sake of conciseness. The system (2.5) was preferred over least squares systems and over quasi-interpolating systems because of the better accuracy of its solutions. Once the $\boldsymbol{c_i}$ are calculated, the curve $t \mapsto \boldsymbol{c(t)}$ is entirely defined. Then the error (2.4) is simply calculated using the norm deduced from the inner product, that is $|f|^2 =< f, f >$.

3. The heuristic knots placement method

The purpose is to set the $m - k + 1$ inner knots $u_k, \dots, u_m$ over $]0, 1[$ such that they form a non-decreasing sequence of real numbers where at most $k - 1$ consecutive values can be identical. Common good sense suggests that the density of the knots should be proportional to the local level of irregularity of the curve L. In high curvature parts of the curve or in the vicinity of cusps the density of knots should increase. To achieve this, we build a continuous and increasing function $t \mapsto V(t), [0, 1] \to [0, 1]$ of which instantaneous slope aims to be proportional to instantaneous irregularity of the curve L.

3.1. Building the distribution knot function $t \mapsto V(t)$

Clearly, V defines a bijection from $[0, 1]$ to $[0, 1]$. Then, to compute the knots, we uniformly spread $m - k + 1$ values $v_k, \dots, v_m$ along $]0, 1[$ and we get the knots by use of the inverse of V. That is,

$$u_i = V^{-1}(v_i) \quad \forall i = k, \dots, m.$$

V^{-1} is used to apply a distortion of $[0, 1]$. It "squeezes" the knots on highly irregular parts of the underlying curve L. If L is a straight line, the distribution is uniform. Let v be a continuous and non identically null function $t \mapsto v(t), [0, 1] \to [0, +\infty[$ of which instantaneous value is proportional to the magnitude of instantaneous irregularity of L. Let $I = \int_0^1 v(t)\mathrm{d}t$ then we define

$$V(x) = \frac{\int_0^x v(t)\mathrm{d}t}{I}$$

Clearly, $x \mapsto V(x)$ verifies all the desired properties. v is built by combination of criteria extracted from successive derivatives of L with respect to time. So we first need to extrapolate derivatives of any order from L.

3.2. Calculating successive derivatives of L

Let $t \mapsto \boldsymbol{q(t)}$ be a such extrapolated derivative of some order of L. We assume that q is defined by interpolating a discrete and dense cloud of points $Q = \{(t_0, \boldsymbol{Q_0}), \dots, (t_n, \boldsymbol{Q_n})\}$ with $n \leq N$ where the t_i denote the time positions and the $\boldsymbol{Q_i}$ the corresponding space positions, this hypothesis holds for L. Then, analogously, we define the derivative $t \mapsto \dot{\boldsymbol{q}}(t)$ by interpolating a cloud of points $R = \{(s_0, \boldsymbol{R_0}), \dots, (s_{n-1}, \boldsymbol{R_{n-1}})\}$ which is defined as follows :

$$\begin{array}{lllll} \text{time position} & : & s_i & = & \frac{t_{i+1}+t_i}{2} \quad \forall i = 0, \dots, n-1 \\ \text{space position} & : & \boldsymbol{R_i} & = & \frac{\boldsymbol{Q_{i+1}}-\boldsymbol{Q_i}}{t_{i+1}-t_i} \quad \forall i = 0, \dots, n-1 \end{array} \tag{3.1}$$

Starting from L and recursively applied, this discrete differentiation provides the successive derivatives of L. The accuracy of the results is quite good thanks to the initial hypothesis of high density cloud. Throughout the iterations, a margin grows on both endpoints of the curve but the density remains constant.

3.3. Angular variations of the successive derivatives

Again, let $t \mapsto \boldsymbol{q(t)}$ be a discrete derivative of some order of L defined by the cloud $Q = \{(t_0, \boldsymbol{Q_0}), \ldots, (t_n, \boldsymbol{Q_n})\}$. Then we can define the discrete angular variation $t \mapsto \alpha(t)$ of q by interpolating a cloud of points $A = \{(s_0, a_0), \ldots, (s_{n-1}, a_{n-1})\}$ which is defined as follows :

$$\begin{array}{lllll} \text{time position} & : & s_i & = & \frac{t_{i+1}+t_i}{2} \quad \forall i = 0, \ldots, n-1 \\ \text{space position} & : & a_i & = & \frac{|angle(Q_i, Q_{i+1})|}{t_{i+1}-t_i} \quad \forall i = 0, \ldots, n-1 \end{array} \tag{3.2}$$

In the following, when q is the i^{th} derivative of L, $t \mapsto \alpha(t)$ is denoted $t \mapsto \alpha_i(t)$. We can give an intuitive meaning of $t \mapsto \alpha_1(t)$: if L is smooth and does not contain any inflexion point then $\frac{\int_0^1 \alpha_1(t)\mathrm{d}t}{2\pi}$ is the number of loops of the curve L. If L is a straight line, then $t \mapsto \alpha_1(t)$ is identically null.

3.4. Criteria

Any of the following functions constitutes a good candidate to represent $v(t)$:

- norm of the discrete i^{th} derivative : $t \mapsto \left|\frac{\mathrm{d}^i}{\mathrm{d}t^i}\boldsymbol{L}(t)\right| \quad \forall i \geq 1$
- discrete angular variations of the i^{th} derivative : $t \mapsto \alpha_i(t) \quad \forall i \geq 1$
- unsigned curvature :$t \mapsto |\rho(t)|$

For any hairpin turn, cusp, gap and more generally any irregularity on L can be detected by a change of magnitude of one or several of those functions. In the following, those functions are referred to as criteria. Let us notice that in [7, 8], the authors use a similar approach but only by considering the third criterion.

3.5. Blending the criteria

To take more benefit from the above criteria it is useful to blend together two or more of them since they can show different kinds of irregularities in different parts of the curve. Without loss of generality, let us detail how we blend two of them : Let $t \mapsto a(t)$ and $t \mapsto b(t)$ be two criteria respectively defined by the two time-space clouds $\{(t_0, a_0), \ldots, (t_n, a_n)\}$ and $\{(t_0, b_0), \ldots, (t_n, b_n)\}$. Since they can be resampled, we can always suppose that both clouds have identical time sequences. We blend a and b by making local convex linear combinations. The operation produces the time-space cloud $\{(t_0, r_0), \ldots, (t_n, r_n)\}$ defined as follows :

$$r_i = \alpha_i \frac{a_i}{\overline{a}} + \beta_i \frac{b_i}{\overline{b}} \quad \forall i = 0 \ldots n \tag{3.3}$$

where :

$$\alpha_i = \frac{\mathcal{A}_i}{\mathcal{A}_i + \mathcal{B}_i} \quad \beta_i = \frac{\mathcal{B}_i}{\mathcal{A}_i + \mathcal{B}_i} \quad \mathcal{A}_i = \frac{|a_i - \overline{a}|}{\sigma_a} \quad \mathcal{B}_i = \frac{\left|b_i - \overline{b}\right|}{\sigma_b}$$

with $\overline{a}$, $\overline{b}$ respectively denoting the population means of $\{a_0, \ldots, a_n\}$ and $\{b_0, \ldots, b_n\}$ and σ_a, σ_b respectively denoting the standard deviations of $\{a_0, \ldots, a_n\}$ and $\{b_0, \ldots, b_n\}$. Equation (3.3) was found empirically, it is motivated by two facts : The farther a coefficient a_i, (resp.b_i) deviates from the mean $\overline{a}$, (resp. $\overline{b}$), the more it means an irregularity. Therefore, its influence on the result should be proportional to this deviation. This influence should be local. Obviously, the purpose of α_i and β_i is only to perform a convex linear combination and the normalization of the coefficients, i.e. the divisions by the means is meant to give all the criteria an equal influence on the resulting blended criterion, no matter their magnitude. Eventually, the function $t \mapsto v(t)$ is built as a continuous function by interpolation of the time-space cloud $\{(t_0, r_0), \ldots, (t_n, r_n)\}$.

Experimentally, as criteria, we used euclidean norms of derivatives from order one up to order 5, angular variations from order one up to 5, plus the curvature function which was calculated using the classical formula :

$$\rho(t) = \frac{det(\dot{\boldsymbol{L}}(t), \ddot{\boldsymbol{L}}(t))}{|\dot{\boldsymbol{L}}(t)|^3}.$$

Table 1: **Test functions used**

$$f_1(t) = \frac{90}{1 + e^{-100(t-0.4)}} \quad t \in [0,1]$$

$$f_2(t) = \begin{cases} \dfrac{1}{0.01 + (t-0.3)^2} & 0 \le t < 0.6 \\ \dfrac{1}{0.015 + (t-0.65)^2} & 0.6 \le t \le 1 \end{cases}$$

$$f_3(t) = \frac{100}{e^{|10t-5|}} + \frac{(10t-5)^5}{500} \quad t \in [0,1]$$

$$f_4(t) = \sin(t) + 2e^{-30t^2} \quad t \in [-2,2]$$

$$f_5(t) = \sin(2t) + 2e^{-16t^2} + 2 \quad t \in [-2,2]$$

$$f_6(t) = \begin{cases} 4t^2(3-4t) & 0 \le t < 0.5 \\ \dfrac{4}{3}t(4t^2 - 10t + 7) - \dfrac{3}{2} & 0.5 \le t \le 0.75 \\ \dfrac{16}{3}t(t-1)^2 & 0.75 \le t \le 1 \end{cases}$$

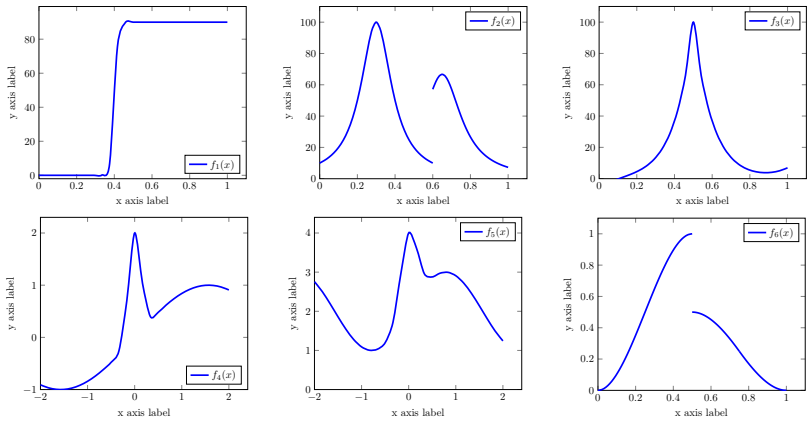

Table 2: Curves corresponding to the six functions $f_1, \ldots, f_6$

We blent from one to three criteria together. The maximum number of functions $t \mapsto v(t)$ built this way is : $\binom{11}{1} + \binom{11}{2} + \binom{11}{3} = 231$. Each one of the functions was used to provide a vector of inner knots which is an input for the fitness function F. Among the population of inner knot vectors produced this way, the best inner knot vector $(u_k, \ldots, u_m)$ was then found by selecting the one yielding the smallest value of F. This knot vector was finally refined by a gradient descent.

4. Experimental results

All development was done by the authors in C++11. The integrals involved in the different calculus were made using trapezoidal rule. Thanks to C++ features, the matrix calculations were easily performed by massive use of operators overloading, template entities and functors (for details, see [20]). All computations were made on a laptop equipped with a processor Intel core i7-8550U CPU. Our method has been compared to three other methods, two deterministic and a stochastic one. The first one consists of launching a gradient descent starting from a uniform distribution of the knots. The second one is the Particle Swarm Optimization algorithm described in [1] and the third is the heuristic algorithm described in [2].

4.1. Implementation details of the comparison methods

The **gradient descent method** was applied to the fitness function F (2.4). The partial derivatives of F are not analytically known (let alone F has not been proved to be differentiable), therefore they were computed using the classical approximating formula :

$$\frac{\partial F}{\partial u_i} \simeq \frac{F(u_k, \ldots, u_{i-1}, u_i + h, u_{i+1}, \ldots, u_m) - F(u_k, \ldots, u_m)}{h} \quad \forall i = k, \ldots, m \tag{4.1}$$

Table 3: Error results of our algorithm, gradient descent, PSO algorithm and Park method for the six noisy functions, w and the handwritten words "@", "blue", "love", "key" and treble clef.

curve	m	**blending criteria**	**Relative Errors**			
			our algorithm	**gradient descent**	**PSO algorithm**[1]	**Park algorithm**[2]
f_1	7	d_2, d_5	2.4×10^{-5}	3.1×10^{-3}	6.3×10^{-5}	1.9×10^{-4}
f_2	13	d_2, α_2, d_5	6.8×10^{-5}	2.7×10^{-4}	8.7×10^{-4}	8.3×10^{-4}
f_3	8	d_2, α_4, ρ	4.9×10^{-5}	1.1×10^{-4}	$2. \times 10^{-4}$	$4. \times 10^{-4}$
f_4	11	d_1, d_3, ρ	2.52×10^{-5}	4.6×10^{-4}	4.64×10^{-5}	1.5×10^{-4}
f_5	9	d_1, α_1, d_3	4.34×10^{-5}	6.83×10^{-5}	5.25×10^{-5}	1.7×10^{-4}
f_6	9	d_3	4.2×10^{-5}	4.47×10^{-5}	6.66×10^{-5}	9.94×10^{-5}
w	17	d_2	8.79×10^{-6}	5.18×10^{-5}	2.63×10^{-5}	1.79×10^{-5}
@	12	d_2, α_4	6.6×10^{-4}	9.5×10^{-4}	$8. \times 10^{-4}$	2.88×10^{-3}
blue	25	α_4	4.2×10^{-4}	$4. \times 10^{-4}$	7.2×10^{-4}	1.1×10^{-3}
love	25	d_1, α_1, α_3	$2. \times 10^{-4}$	1.8×10^{-4}	2.6×10^{-4}	6.8×10^{-4}
key	23	d_2, α_1, α_4	9.38×10^{-5}	1.3×10^{-4}	3.1×10^{-4}	3.1×10^{-4}
treble clef	13	α_1, d_5, ρ	5.9×10^{-4}	6.8×10^{-4}	7.9×10^{-4}	2.1×10^{-3}

Table 4: CPU time results of our algorithm, gradient descent, PSO algorithm and Park method for the six noisy functions, w and the handwritten words "@", "blue", "love", "key" and treble clef.

curve	m	**CPU time (in s.)**			
		our algorithm	**gradient descent**	**PSO algorithm**[1]	**Park algorithm**[2]
f_1	7	2.65	2.45	5.2	0.017
f_2	13	2.9	2.8	6.2	0.02
f_3	8	2.7	2.2	4.9	0.028
f_4	11	3.3	1.8	6.21	0.031
f_5	9	2.88	2.67	5.84	0.015
f_6	9	2.6	2.52	5.87	0.017
w	17	3.97	3.77	7.51	0.037
@	12	7.45	6.84	10.8	0.076
blue	25	38.9	36.1	48	0.93
love	25	30	28	39	0.5
key	23	10	9.3	15.7	0.093
treble clef	13	12	11	16	0.27

with h being a small positive real number. The starting point used to initiate the gradient descent method was set to the uniform distribution.

The **PSO algorithm** used was the one described in [1], we also used the same parameters. Throughout the iterations, the sequences $(u_k, \ldots, u_m)$ have to remain non decreasing and confined to the interval $[0, 1]$. Both conditions are hardly met by the algorithm though this issue was not discussed in the original paper. So we made our own adjustments to fix it. After each iteration : every knot u is replaced by $u - \lfloor u \rfloor$ where $\lfloor . \rfloor$ is the floor function and all the knots are sorted to ascending order. The same arrangement was done on the gradient descent method.

The Park method is the heuristic algorithm described in [2]. The method is based on the idea of dominant points which are a subset of the initial given data points P_i. Then the knots are computed as mean values of $k-1$ consecutive dominant points. The algorithm consists in selecting the $m+1$ best dominant points. The algorithm is composed of two steps. The first step determines the dominant *seed* points set. It consists of a few pre-selected data points which maximize the curvature. These points are used to build a first coarse B-spline approximation of the curve L. The second step consists in repeated locally refining the approximation by selection of new dominant points. For both steps of the algorithm (seed points initiation and adaptive refinement) the instantaneous curvature was calculated the same way we did in our method (i.e. by use of the two first discrete derivatives of $t \mapsto \boldsymbol{L}(t)$) since the results are very accurate.

For our algorithm, the knot vector $(u_k, \ldots, u_m)$ was found following the method described in subsection 3.5. This knot vector was then refined by a gradient descent. Quite good results were obtained with 10 to 20 iterations. The accuracy of the four methods (Gradient, PSO, Park and our algorithm) was simply measured by the value of the fitness function F which is the squared relative error between the raw sampled curve and the resulting B-Spline curve. It is therefore based on the norm induced by the inner product on

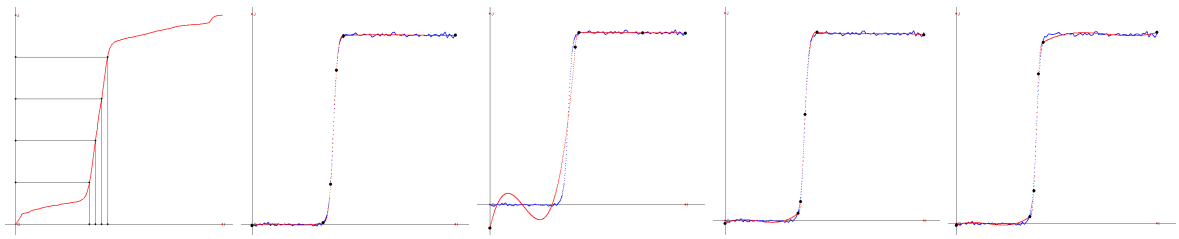

Figure 1: Approximation curves (red) of the original noisy curve f_1 (blue) produced by the different methods. From left to right : Knot placement of 4 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

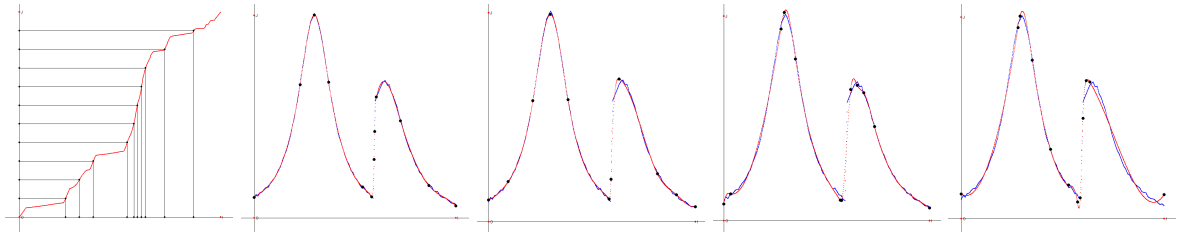

Figure 2: Approximation curves (red) of the original noisy curve f_2 (blue) produced by the different methods. From left to right : Knot placement of 10 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

functions. That is, the accuracy of a stochastic method cannot easily be defined since its results vary with every execution.

4.2. Benchmark

Half of the benchmark of sampled curves used to make the comparisons is the one provided in [1]. It consists in a series of six non parametric functions denoted f_1 to f_6 in the following. First the tests were performed on the smooth functions then on noisy versions of them. A cubic B-Spline curve made of 19 control points (i.e. $m = 18$) was also added to the benchmark. It is shaped as a handwritten "w" with an additional angular point. It has been sampled then been made noisy in the same way as were $f_1, \ldots, f_6$. In the following, this curve is simply denoted as w. To complete this benchmark, five handwritten words "@", "blue", "love", "key" and treble clef- therefore naturally noisy - were added. They were obtained from a software developed by the authors and running on an Android device. The clouds respectively contain 185, 418, 355, 175 and 250 points. So, altogether, we ran 72 tests (18 curves $\times$ 4 methods). The six functions $f_1, \ldots, f_6$ are summarized in table 1, their corresponding curves are drawn on table 2. They were uniformly sampled one hundred times over their domains. Then they were made noisy by adding a gaussian noise $N(0, \sigma^2)$ produced by use of Box-Muller method [5]. The different values of σ for $f_1, \ldots, f_6$ and w were respectively 0.005, 0.005, 0.005, 0.01, 0.02, 0.005 and 0.005.

4.3. Results and discussion

Table 3 shows the results of the four methods applied to the six noisy functions $f_1, \ldots, f_6$, w and the handwritten curves "@", "blue", "love", "key" and treble clef. In table 3, d_i,α_i, ρ respectively denote the norm of the i^{th} derivative of L, the angular variation of the i^{th} derivative of L, the unsigned curvature of L. For those 12 curves, the applicant pool of criteria to blend from were euclidean norms of derivatives from order one up to order five, angular variations from order one to five, plus the curvature function. Widening the range of derivatives would yield more accuracy but would be too time-consuming. Eventually, order five was an acceptable trade-off between time and accuracy. Since the raw curves are noisy, the criteria have been extracted from smoothed versions of the initial curves. This was made with a band diagonal matrix.

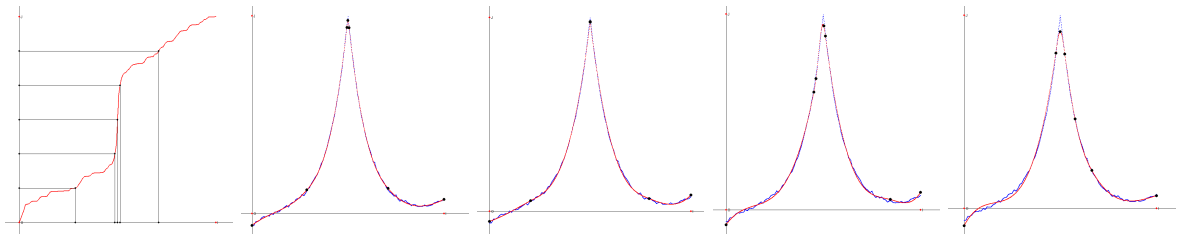

Figure 3: Approximation curves (red) of the original noisy curve f_3 (blue) produced by the different methods. From left to right : Knot placement of 5 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

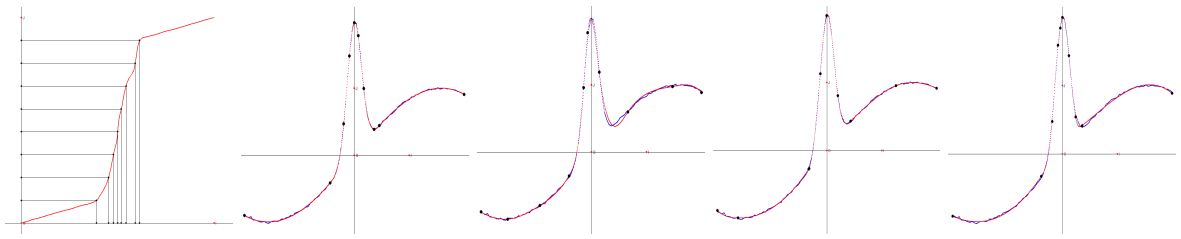

Figure 4: Approximation curves (red) of the original noisy curve f_4 (blue) produced by the different methods. From left to right : Knot placement of 8 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

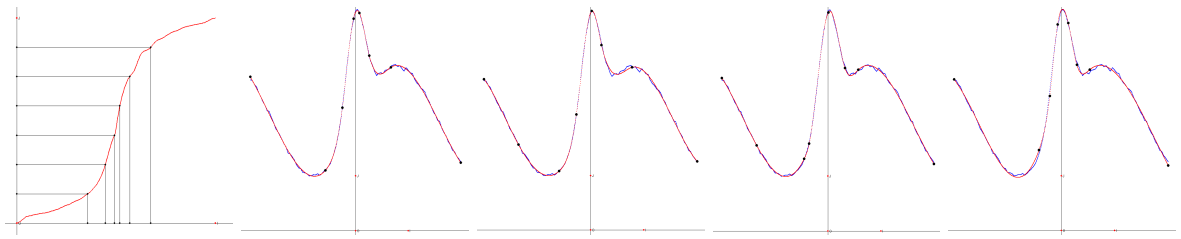

Figure 5: Approximation curves (red) of the original noisy curve f_5 (blue) produced by the different methods. From left to right : Knot placement of 6 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

Figure 6: Approximation curves (red) of the original noisy curve f_6 (blue) produced by the different methods. From left to right : Knot placement of 6 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

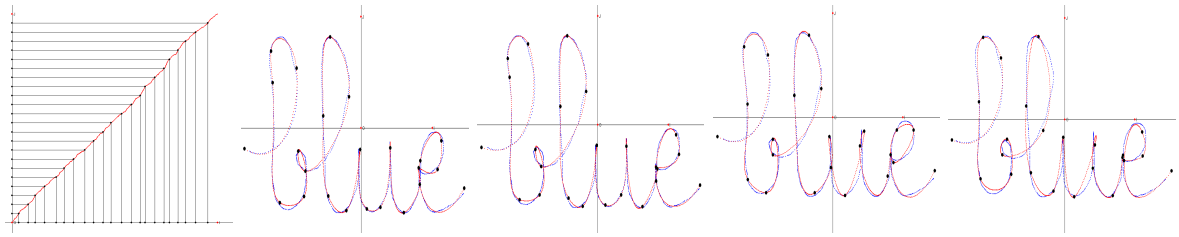

Figure 7: Approximation curves (red) of the original curve *blue* (blue) produced by the different methods. From left to right : Knot placement of 22 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

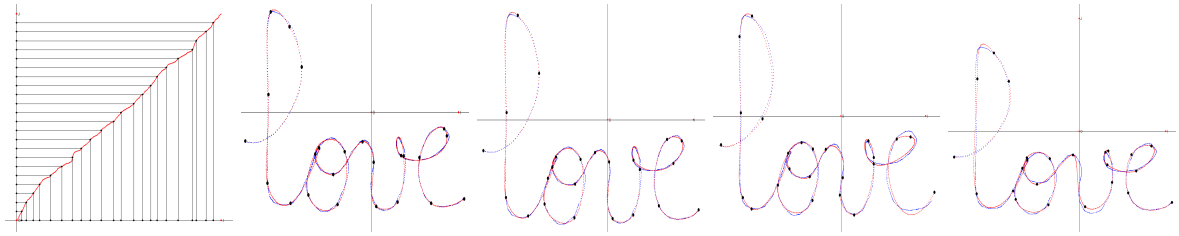

Figure 8: Approximation curves (red) of the original curve *love* (blue) produced by the different methods. From left to right : Knot placement of 22 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

In the following, we compare the four methods in terms of relative errors then in terms of CPU time, Table 3 summarizes the comparison accuracy results of our algorithm, the descent method, the particle swarm algorithm and Park algorithm. The third column in these tables shows which combination of criteria produced the best inner-knot vector for our algorithm. Table 3 points out that for the considered examples, the proposed algorithm gives the best results.

The comparison against PSO algorithm shows that the results are slightly better with handwritten words and twice as better with the synthetic curves. The PSO algorithm yields goods results either the curves are synthetic or handwritten. Unfortunately, it sometimes crashes because likely the Schoenberg-Whitney condition is not satisfied. Let us note that it is hard to compare deterministic methods with stochastic methods. How many attempts should we do with the latter ? If we allow more attempts with stochastic methods then deterministic methods should be allowed more evaluations too.

The comparison against Gradient descent shows that the results are similar with handwritten words and twice as better with the synthetic curves. The experimental results show that Gradient descent sometimes misses the global minimum and converges to a local minimum. For example see the curve f_1 (fig.1).

The comparison against Park algorithm shows that the improvement factor accuracy is ranging from 2 to 10. Following the authors advice, the value of r which measures the compromise between curvature and length was set to 0.8 for all examples. Though the authors also suggested to set the number of dominant seed points to 10, we set it to 4 or 6 since small values usually yielded more accurate results. Let us notice that Park method is very sensitive to the number of dominant seed points. The error magnitude can grow by a factor of two when choosing a wrong number of seed points. For example, the experimental results show that with the curve @, when $m = 9$ and $r = 0.8$, the error grows from 0.0137 to 0.268 when the seed number varies from 4 to 8. As a general rule, with Park method, one must take more internal knots to achieve the same error level as our method does.

To make a fair comparison of the four methods, we allow them to make the same amount of computations of the value of the fitness function F which is, by far, the most CPU time-costly step of each algorithm. For the above tests, the number of computations of F was always set between 200 and 700. That is, this

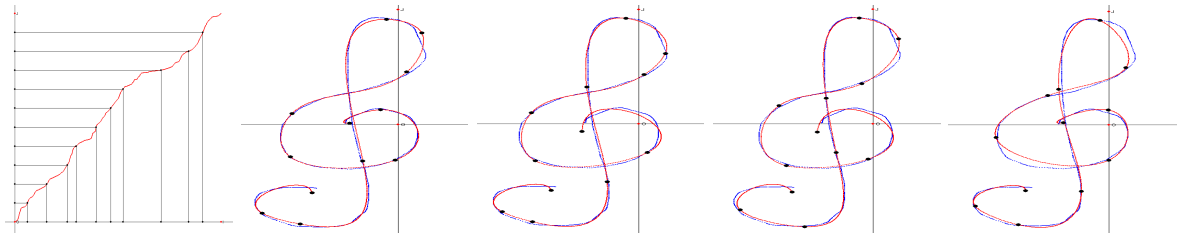

Figure 9: Approximation curves (red) of the original curve *trebleclef* (blue) produced by the different methods. From left to right : Knot placement of 10 inner knots using Distribution function $V(x)$, Our algorithm, Gradient descent, PSO algorithm, Park algorithm

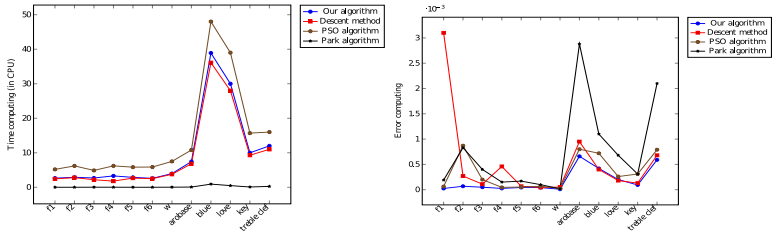

Figure 10: (left) CPU time and (right) Relative errors results of the four methods for the twelve curves.

is meaningful for Gradient descent, PSO and our algorithm which are time-consuming methods but not for Park method which does not require an evaluation of F and therefore is instantaneous compared to the three others (see table 4).

Table 4 and figure 10 show that Park method is the fastest method. As it was predictable, PSO algorithm is the slowest method. Our algorithm and Gradient descent show similar results. Park method is instantaneous but if one needs several attempts to manually tune the values of parameters and number of seed points, one looses all the benefit of the immediacy of the method. Obviously, our algorithm is slower than Park method since it exhaustively and automatically examines combinations of many geometric criteria.

In both algorithms, the computed knots depend on local geometric properties of the considered data curve. Park method empirically selects two criteria : length and curvature. On the contrary, our algorithm tries all combinations of many geometrical criteria which include the norm of the first derivative (which is correlated to length) and curvature. That means that Park algorithm can be seen as a particular case of our algorithm. One can note that Park method is not well suited to discontinuous curves as shown on curves f_2 (see fig. 2) and f_6 (see fig. 6), neither to non smooth functions as shown in f_3 (see fig. 3). This could be explained by the fact that Park method is based on a discrete model, therefore it does not take into account the information in between two consecutive data points. It also means that the algorithm depends on the sample rate since the increase of it would reduce this problem. For the same reason, the accuracy of Park method is very sensitive to the number of seed points. The sensitivity to the sample rate is less obvious in our method since our model is continuous. Since our algorithm distorts an initially uniform distribution, it is very unlikely that it would result in an ill-conditioned inner-knot vector. This robust behaviour shows as well with gradient-descent method and Park method but not with PSO algorithm. Our algorithm does not need to manually set any parameters. It automatically selects the best parameters when constructing the function v. The main feature of our algorithm is that it never degenerates : It is always reliable whereas the three others are not.

5. Conclusion

In this paper, we propose an heuristic method for knot placement in B-spline approximation. It is based on blending geometric characteristics of the data. It has been successfully compared to three other methods

: Gradient descent, Particle swarm optimization and Park method. The proposed algorithm gives good accuracy results and it is more reliable than the other ones. Unlike the other ones, it is self contained since it does not need to set any parameters. The experimental results show that our algorithm is faster that PSO algorithm, but much slower than Park algorithm which is instantaneous. However, the latter needs more internal knots to achieve the same accuracy as our algorithm does. The main idea has been validated by the tests but some steps of the algorithm might be improved. Our algorithm is sensitive to noise and how to select the relevant criteria is the main issue. We are a bit blind about which criterion to use and which one to rule out. In further work, we shall try to use principal component analysis to see if any general rule appears about the criteria. We also wish to generalize this method to surface approximation.

[1] Gálvez A., Iglesias A., Efficient particle swarm optimization approach for data fitting with free knot B-splines, Computer-Aided Design 43 (2011) 1683-1692.

[2] Park H., Lee J.-H., B-spline curve fitting based on adaptive curve refinement using dominant points, Comput. Aided Des. 39 (6) (2007) 439âĂŞ451.

[3] Boehm W., Inserting new knots into B-spline curves, Computer Aide d Design 12(1980), 199-201.

[4] De Boor C., A Practical Guide to Splines, Springer-Verlag, New York, (1978).

[5] G.E.P Box, M.E. Muller, "A note on the generation of random normal deviates", Annals Math. Stat 29 (1958) 610-61.

[6] Burchard HG. Splines (with optimal knots) are better. Applicable Analysis 3(1974) 309-19.

[7] Razdan A. Knot placement for B-spline curve approximation. In: Technical report. Arizona State University; 1999.

[8] Li, W., Xu, S., Zhao, G., Goh, L.P., Adaptive knot placement in b-spline curve approximation. Comput. Aided Des.37 (8) (2005), 791âĂŞ797.

[9] Goldenthal R. , Bercovier M., Spline curve approximation and design by optimal control over the knots, Computing 72 (2004) 53-64.

[10] Jupp DLB. Approximation to data by splines with free knots. SIAM Journal on Numerical Analysis 15(1978) 328-43.

[11] Rice JR. The approximation of functions, vol. 2. Reading, MA: Addison-Wesley; 1969.

[12] Ülker E. , Arslan A., Automatic knot adjustment using an artificial immune system for B-spline curve approximation. Information Sciences 179 (2009) 1483-1494

[13] Yoshimoto F., Harada T., Yoshimoto Y., Data fitting with a spline using a real coded algorithm. Computer-Aided Design 35 (2003) 751-60.

[14] Yuhua Zhang, Juan Cao, Zhonggui Chen, Xin Li , Xiao-Ming Zeng, B-spline surface fitting with knot position optimization. Computers & Graphics 58(2016)73âĂŞ83.

[15] Hongmei Kang, Falai Chen, Yusheng Li, Jiansong Deng, Zhouwang Yang, Knot calculation for spline fitting via sparse optimization. Computer-Aided Design 58 (2015) 179âĂŞ188.

[16] Jiaqi Luo, Hongmei Kang, Zhouwang Yang, Knot calculation for spline fitting based on the unimodality property. Computer Aided Geometric Design 73 (2019) 54âĂŞ69.

[17] Pascal Laube, Matthias O.Franz, Georg Umlauf, Learnt knot placement in B-spline curve approximation using support vector machines, Computer Aided Geometric Design 62 (2018) 104âĂŞ116

[18] Olga Valenzuela, Blanca Delgado-Marquez, Miguel Pasadas, Evolutionary computation for optimal knots allocation in smoothing splines. Applied Mathematical Modelling 37 (2013) 5851âĂŞ5863.

[19] H. Idais, M. Yasin, M. Pasadas, P. González, Optimal knots allocation in the cubic and bicubic spline interpolation problems. Mathematics and Computers in Simulation 164 (2019) 131âĂŞ145.

[20] Allain A., Jumping into C++, Programming.com edition, 2013.