

UNIVERSITÀ DEGLI STUDI DI TRENTO  
Facoltà di Scienze Matematiche, Fisiche e Naturali



Corso di Laurea Specialistica in Informatica  
Elaborato finale

---

RECONSTRUCTING CURVES FROM SPARSE AND NOISY DATA  
THROUGH REACTIVE SEARCH OPTIMIZATION TECHNIQUES

Relatore:  
Roberto Battiti

Laureando:  
Michele Dallachiesa

Anno Accademico 2008 - 2009

*"La verità è una terra senza sentieri"*

Jiddu Krishnamurti

*Ai miei nonni*

## **Acknowledgements**

My sincerest thanks belong to my father and my mother for the support they provided me through my entire life, in the first place for their commitment to my studies. I would like to express my gratitude to Professor Roberto Battiti. It was a pleasure to write this thesis profiting from his expertise, patience and valuable suggestions. Last but not least, my girlfriend Angela made me a happy person and gave me the extra strength, motivation and love necessary to get things done. And finally, a Cheers! goes out to all you other people I spent great times with in the past and will do so in the future.

## Abstract

In this study, we consider the problem of reconstructing curves from sparse and noisy data by using the B-spline curve model. The main contributions of this work can be summarized as follows. We consider a least squares formulation of fitting B-splines, solved by reactive search optimization (RSO) techniques. We discuss different heuristics to determine adaptively the number and the initial values of control points, an argument not sufficiently covered in the existing literature. We propose also some new adaptive parametrization techniques, showing that a hybrid approach can outperform the basic *centripetal* scheme while preserving a low computational cost.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Problem setting . . . . .	3
1.2	Overview . . . . .	3
<b>2</b>	<b>Curve fitting approaches</b>	<b>4</b>
2.1	Polynomial interpolation . . . . .	4
2.2	Spline interpolation . . . . .	5
2.3	B-spline approximation . . . . .	7
2.3.1	The parametrization problem . . . . .	10
2.3.2	Generation of the knots vector . . . . .	14
<b>3</b>	<b>RASH: The Reactive Affine Shaker</b>	<b>15</b>
3.1	The Reactive Affine Shaker Algorithm . . . . .	16
3.1.1	Motivation and analysis . . . . .	17
3.1.2	RASH pseudo-code . . . . .	20
3.1.3	Termination and repeated runs . . . . .	23
<b>4</b>	<b>B-spline fitting with RASH</b>	<b>25</b>
4.1	Characterization of the optimal solution . . . . .	26
4.2	A simple "goodness of fit" measure . . . . .	27
4.3	Objective function for minimization . . . . .	27
4.4	The optimization process . . . . .	29

<b>5</b>	<b>Evaluation of the optimization process</b>	<b>30</b>
5.1	Generation of test curves . . . . .	31
<b>6</b>	<b>Influence of the initial solution and number of control points</b>	<b>34</b>
6.1	Number of control points . . . . .	35
6.2	Random control points initialization . . . . .	36
6.3	Random subset of data points as control points initialization . . . . .	42
6.4	Uniform control points initialization . . . . .	42
6.5	Reactive choice of number and position of control points . . . . .	44
6.5.1	Slope-based initial solution . . . . .	45
6.5.2	Change-of-direction-based initial solution . . . . .	48
6.5.3	Data preprocessing by sampling and smoothing . . . . .	56
<b>7</b>	<b>The parametrization problem</b>	<b>65</b>
7.1	The global minimum distance parametrization . . . . .	65
7.2	The local minimum distance parametrization . . . . .	66
7.3	The adaptive hybrid parametrization scheme . . . . .	72
<b>8</b>	<b>Conclusions</b>	<b>73</b>
8.1	Future research directions . . . . .	74
<b>A</b>	<b>Some complex examples</b>	<b>76</b>
	<b>References</b>	<b>84</b>

# Chapter 1

## Introduction

Curve fitting is the process of determining a curve that has the best fit to a series of data points. The reconstruction of curves can involve either interpolation, where an exact fit to the data is required, or approximation, in which a "smooth" function is constructed that approximately fits the data. Fitted curves can be used as an aid for data visualization, to infer values of a function where no data are available, and to summarize the relationships among two or more variables. The spectrum of possible applications is wide, ranging from moving object tracking to contours detection in computer vision. In the presence of experimental data points affected by noise, approximation is to prefer to interpolation to filter out the measurement errors. In this study, we consider the problem of reconstructing curves from sparse and noisy data through reactive search optimization (RSO) techniques.

Reactive Search [2] advocates the integration of sub-symbolic machine learning techniques into search heuristics for solving complex optimization problems. The word reactive hints at a ready response to events during the search through an internal online feedback loop for the self-tuning of critical parameters. In our contexts, the self-tuning is related to the adaptation of the search region in the RASH algorithm but also to more specific methods based on the preprocessing of the experimental data points leading to new control points generation methods, to new adaptive parametrization techniques, and

to a hybrid approach that can outperform the basic schemes while preserving a low computational cost.

## 1.1 Problem setting

We consider the problem of reconstructing a curve from a series of  $s+1$  sparse and noisy data points. Given a series of  $s+1$  data points  $d_0, \dots, d_s$ , the goal is to approximate them with a curve, while ensuring some desired properties: The determined curve must be continuous up to the second derivative and pass exactly through the first and last data points  $d_0$  and  $d_s$ . The adopted curve model should be sufficiently powerful to support also complex shapes, as object contours and moving object trajectories.

## 1.2 Overview

The following parts of this work are organized as follows. In Chapter 2, we discuss and compare various methods for curve fitting, namely polynomial interpolation, spline interpolation and B-spline approximation. In Chapter 3, we present related works on B-spline fitting curves and RASH, the optimization scheme we adopt in our proposal. In Chapter 4, we define a simple "goodness of fit" measure and the objective function for minimization, discussing the optimization process in details. The generation process of data points and the evaluation function that measures the quality of fit are presented in Chapter 5. The choice of the number of control points and their initial positioning is a key point for B-spline fitting curve generation. We address these problems in Chapter 6. Then, we propose some adaptive parametrization techniques in Chapter 7. In Chapter 8, we give our conclusions. In Appendix A, we report some examples of fitting B-splines applied to complex shapes.

# Chapter 2

## Curve fitting approaches

Curve fitting is the process of reconstructing a curve that has the best fit to a series of data points, possibly subject to constraints. Curve fitting can involve either interpolation, where an exact fit to the data is required, or approximation, in which a "smooth" function is constructed that approximately fits the  $s + 1$  data points  $d_0, \dots, d_s$ . We will briefly cover the most commonly used approaches, with pros and cons in our specific problem setting.

### 2.1 Polynomial interpolation

Polynomial based approaches serve well due to their simple formulation and their easy implementation [15]. Given some data points, the aim is to find a polynomial which goes exactly through these points. The principle is that through any two points there is a unique line, through any three points a unique quadratic, and so on. The interpolating polynomial of degree at most  $s$  that passes through the  $s + 1$  points  $d_k = (x_k, y_k)$  for  $k = 0, \dots, s$  is given explicitly by the Lagrange's classical formula

$$P(x) = \sum_{k=0}^s y_k L_{s,k}(x) \tag{2.1}$$

where  $L_{s,i}(x)$  is the Lagrange coefficient polynomial

$$L_{s,k}(x) = \frac{(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_s)}{(x_k - x_0) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_s)} \quad (2.2)$$

The equation (2.2) can be also expressed in the more compact form

$$L_{s,k}(x) = \frac{\prod_{j=0, j \neq k}^s (x - x_j)}{\prod_{j=0, j \neq k}^s (x_k - x_j)} \quad (2.3)$$

The resulting polynomial  $P(x)$  goes exactly through the given points. However, this solution suffers from the Runge's phenomenon [16]: The Runge's phenomenon is a problem that occurs when using polynomial interpolation with polynomials of high degree. This situation can arise when trying to interpolate many data points, resulting in oscillations between the interpolated data points. The problem can be avoided by using spline curves which are piecewise polynomials, usually of third degree. Spline interpolation is presented in the next section.

## 2.2 Spline interpolation

Spline interpolation is a form of interpolation where the interpolant is a special type of piecewise polynomial called spline. We will consider here natural cubic spline interpolation, the most widely used [8] [15]. Historically, the motivation to introduce natural cubic spline interpolation is based on the engineer's device used to draw smooth curves through a number of points, as in Figure 2.1. This spline consists of weights attached to a flat surface at the points to be connected. A flexible strip is then bent across each of these weights, resulting in a pleasingly smooth curve.

The mathematical spline is similar in principle. The points, in this case, are numerical data and the weights are the coefficients on the cubic polynomials used to interpolate the data. Given a sequence of  $s + 1$  data points  $d_k = (x_k, y_k)$  for  $k = 0, \dots, s$ , the essential idea is to fit a piecewise function

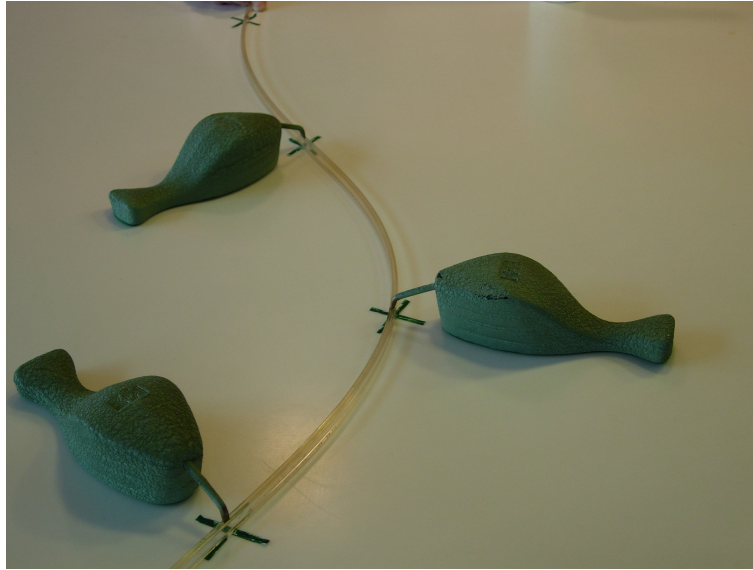


Figure 2.1: Physical spline

of the form

$$S(x) = \begin{cases} S_0(x) & x \in [x_0, x_1] \\ S_1(x) & x \in [x_1, x_2] \\ \vdots & \vdots \\ S_{s-1}(x) & x \in [x_{s-1}, x_s] \end{cases} \quad (2.4)$$

For each  $S_i(x)$ ,  $i = 0, \dots, s - 1$ , we require four conditions:

1. The interpolating property,  $S(x_k) = y_k$
2. The splines to join up,  $S_{i-1}(x_k) = S_i(x_k)$
3. Once continuously differentiable,  $S'_{i-1}(x_k) = S'_i(x_k)$
4. Twice continuously differentiable,  $S''_{i-1}(x_k) = S''_i(x_k)$

For the  $s$  cubic polynomials comprising  $S$ , we need to determine  $4s$  conditions since for one polynomial of degree three, there are four parameters. The interpolating property gives us  $s + 1$  conditions, while the continuity conditions on the interior data points  $x_1, \dots, x_{s-1}$  give us  $3(s - 1)$  constraints

each. Summing up, we have  $(s + 1) + 3(s - 1) = 4s - 2$  conditions. We require two other conditions. This choice characterizes the natural splines, which add those two constraints:  $S''(x_0) = S''(x_s) = 0$ . The natural cubic spline is approximately the same curve as created by the spline device used by engineers in the past. In order to compute the coefficients, we need to solve the linear system of these  $4s$  equations. Splines can model any sufficiently smooth curve defined in the form  $y = f(x)$ , but they cannot describe curves which have multiple values for  $y$  given  $x$ . This limitation can be removed by introducing B-splines, special splines which are also parametric curves. Approximating B-spline curves are presented in the next section.

## 2.3 B-spline approximation

General B-splines are parametric curves that provide a highly versatile approach to describe curves in many fields, including computer graphics. B-splines, as splines, consist of sections of polynomial curves connected at some points called knots with some continuity constraints. In other words, B-splines are a linear combination of basis functions, weighted through the control points. They do not pass through their control points, contrary to local interpolating splines. An example of B-spline is reported in Figure 2.2, where the vertical lines identify the sections of polynomial curves.

The polynomials of a given B-spline all have the same degree, which is the degree of the B-spline, joining at some points, called knots. The most used B-splines consist of cubic segments, and are commonly identified as cubic B-splines. A B-spline curve can be represented in the form

$$C(u) = \sum_{i=0}^n N_{i,p}(u)P_i \quad (2.5)$$

Where  $N_{i,p}$  is the B-spline basis function of degree  $p$  and  $P_i$  is the  $i$ th control point. B-splines can be defined completely through a set of  $n + 1$  control points, a knots vector of  $m + 1$  values (or, knots vector) and a degree  $p$ . Note

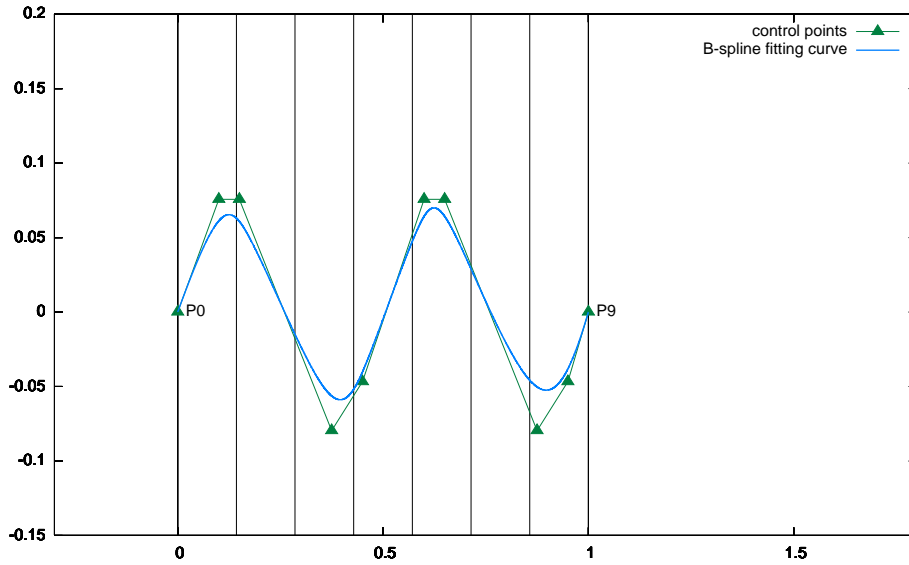


Figure 2.2: B-spline curve with explicit knots distribution

that  $n$ ,  $m$  and  $p$  must satisfy

$$m = n + p + 1 \quad (2.6)$$

More precisely, if we want to define a B-spline curve with  $n + 1$  control points, we have to supply  $n + p + 2$  knots  $u_0, \dots, u_{n+p+1}$ . The point on the curve that corresponds to a knot  $u_i$ ,  $C(u_i)$ , is referred to as a knot point. Hence, the knot points divide a B-spline curve into curve segments, each of which is defined on a knot span. This division is explicitly reported in Figure 2.2. To change the shape of a B-spline curve, we have to modify control points, knots or the degree of the basis functions. Note also that to add a control point to a B-spline, we must add also a new knot. Vice versa, if we want to reduce the number of control points, we must reduce also the number of knots. B-spline basis functions can be defined in a numerically stable way by the de Boor algorithm

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+1} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i-1,p-1}(u) \quad (2.8)$$

where parameter  $u$  ranges in the interval  $[0, 1]$ ,  $u_0, \dots, u_m$  are the knots and  $p$  is the degree of the basis functions. The above is usually referred to as the Cox-de Boor recursion formula [8]. An efficient implementation of the algorithm should consider a table of precomputed values for  $N_{i,p}(u)$ . Note that  $N_{i,p}(u)$  is non-zero on interval  $[u_i, u_{i+p+1})$ . If  $u$  is not in this interval,  $N_{i,p}(u)P_i$  has no effect in computing  $C(u)$  since  $N_{i,p}(u)$  is zero. Usually, control points don't pass through the B-spline curve, but we may be interested in forcing this constraint for the first and last control point. This can be achieved with the so called clamped B-splines, by "clamping" the first  $p+1$  and last  $p+1$  knots (i.e.,  $u_0 = u_1 = \dots = u_p$  and  $u_{m-p} = u_{m-p+1} = \dots = u_m$ ) and by initializing the first and last control points to the first and last data points, respectively  $P_0 = d_0, P_n = d_s$ . B-spline curves have two important properties. In the following we shall assume a B-spline curve  $C(u)$  of degree  $p$  that is defined by  $n+1$  control points and a knots vector  $U = \{u_0, u_1, \dots, u_m\}$  with the first  $p+1$  and last  $p+1$  knots "clamped".

- Local modification: Changing the position of control point  $P_i$  only affects the curve  $C(u)$  on interval  $[u_i, u_{i+p+1})$ . Recall that  $N_{i,p}(u)$  is non-zero on interval  $[u_i, u_{i+p+1})$ . If  $P_i$  changes its position,  $N_{i,p}(u)P_i$  is changed and consequently  $C(u)$  is changed only if  $u$  is in the interval  $[u_i, u_{i+p+1})$ .
- Affine invariance: If an affine transformation is applied to a B-spline curve, the result can be constructed from the affine images of its control points. In other words, if we want to apply a geometric or even affine transformation to a B-spline curve, it is sufficient to apply this modification to its control points.

We have now defined precisely what a B-spline is. Now, we consider how to use B-splines to fit a series of data points. We consider approximating

B-splines instead of interpolating B-splines because the latter ones memorize exactly the data points, which in our particular scenario may be sparse and noisy. In this context, our desire is to derive a model curve explaining the measured points but also "generalizing" well. Fitting a set of data points with a B-spline curve through approximation can be seen as the process of relaxing the strict requirement that the curve must pass through all data points. Except for the first and last data points, the curve does not have to pass strictly through any other point. To measure how well a curve approximates the given data points, the concept of error distance is used. The error distance measure is the distance between a data point and its "corresponding" point on the B-spline curve. The objective of the approximation is to minimize the sum of the error distances. Finding the "corresponding" point on the B-spline curve which correspond to the data points is a procedure referred to as parametrization. Once the parameters and the knots vector are determined, the remaining variables of the B-spline curve are its control points values, which may be determined through optimization techniques. We will discuss separately this problem in the next Chapter. In the next Sections, we present the classical parametrization schemes and the generation of the knots vector.

### 2.3.1 The parametrization problem

As we said, we need to associate each data point to a point of the B-spline fitting curve in order to determine the approximation error for this point. More formally, the input to a B-spline approximation algorithm usually consists of a set of data points. Thus, the first step is to find a set of parameters that can correspond to these points at certain values. Let  $d_0, \dots, d_s$  be a given sequence of ordered data points derived from the curve we want to fit. Then,  $s + 1$  parameters  $t_0, \dots, t_s$  in the domain of the curve must be found so that data point  $d_k$  corresponds to parameter  $t_k$  for  $k = 0, \dots, s$ . This means that if  $C(u)$  is a curve that passes through all data points in the given order, then we have  $d_k = C(t_k)$  for all  $0 \leq k \leq s$ . This happens with interpolating curves, while with approximating curves the difference is used as a measure of the er-

ror to minimize. There are infinite number of possibilities for selecting these parameters. For example, we can evenly divide the domain, or randomly pick  $s + 1$  values from the domain. However, poorly chosen parameters can cause unpredictable results: An improper data parametrization may considerably compromise the quality and efficiency of approximation. We now present three classical parametrization schemes.

- The *uniformly spaced* method is the simplest parameter selection scheme. Let's assume that the domain, as usual, is  $[0, 1]$  and  $s + 1$  uniformly spaced parameters are required. The first and the last parameters must be 0 and 1 because we want the curve to pass through the first and the last data points. Therefore, we have  $t_0 = 0$  and  $t_s = 1$ . Since  $s+1$  points divide the interval  $[0, 1]$  into  $s$  subintervals evenly, each of which must be of length  $1/s$ , the division points are  $0, 1/s, 2/s, 3/s, \dots, (s - 1)/s$  and 1. Thus, we have

$$t_i = \begin{cases} 0 & k = 0 \\ \frac{k}{s} & 1 \leq k \leq s - 1 \\ 1 & k = s \end{cases} \quad (2.9)$$

While the *uniformly spaced* method is simple, it is known to generate some unpleasant results. For example, when the data points are not uniformly spaced, using uniformly spaced parameters could generate erratic shapes such as big bulges, sharp peaks and loops.

In Figure 2.3, we report a parametrization example, where the  $\sin(x)$  curve in interval  $[0, 4\pi]$  is derived from 40 uniformly spaced data points. The corresponding 40 parameters on the B-spline curve are chosen uniformly in interval  $[0, 1]$ . The arrows from the data points to the B-spline curve represent the association between data points and the corresponding points on the spline.

- The *chord-length* method derives from the observation that if an interpolating curve follows very closely to the data points, the length of the

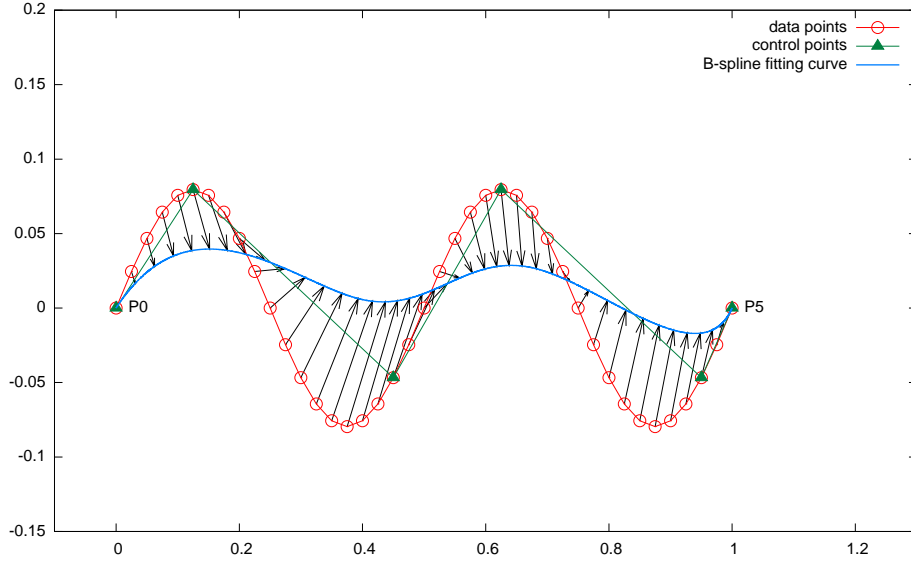


Figure 2.3: Uniform parameters of a B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 4\pi]$ , derived from 40 uniformly spaced data points

curve segment between two adjacent data points would be very close to the length of the chord of these two data points, and the length of the interpolating curve would also be very close to the total length of the polyline constructed connecting adjacent data points. Therefore, if the domain is subdivided according to the distribution of the chord lengths, the parameters will be an approximation of the arc-length parameterization, a common parametrization approach with parametric curves. Suppose the data points are  $d_k$ , with  $k = 0, \dots, s$ . The length between  $d_{k-1}$  and  $d_k$  is  $|d_k - d_{k-1}|$ , and the length of data points polyline is the sum of the lengths of these chords:

$$L = \sum_{k=1}^s |d_k - d_{k-1}| \quad (2.10)$$

Therefore, the ratio of the chord length from data point  $d_0$  to data point  $d_k$ , denoted as  $L_k$ , over the length of the data points polyline, is

$$L_k = \frac{\sum_{k=1}^s |d_k - d_{k-1}|}{L} \quad (2.11)$$

If we prefer to have an arc-length parameterization of the interpolating curve, the domain has to be divided according to the ratio  $L_k$ . More precisely, if the domain is  $[0, 1]$ , then parameter  $t_k$  should be located at the value of  $L_k$ :

$$t_k = \begin{cases} 0 & k = 0 \\ L_k & 1 \leq k \leq s - 1 \\ 1 & k = s \end{cases} \quad (2.12)$$

The *chord-length* method is widely used and usually performs well.

- The *centripetal* method [11] is an extension of the *chord-length* method: The distance between two adjacent data points is measured by  $|d_k - d_{k-1}|^a$  rather than the conventional  $|d_k - d_{k-1}|$ . We can adjust the equations of the *chord-length* method as follows:

$$L = \sum_{k=1}^s |d_k - d_{k-1}|^a \quad (2.13)$$

$$L_k = \frac{\sum_{i=1}^s |d_k - d_{k-1}|^a}{L} \quad (2.14)$$

$$t_i = \begin{cases} 0 & k = 0 \\ L_k & 1 \leq k \leq s - 1 \\ 1 & k = s \end{cases} \quad (2.15)$$

If  $a = 1$ , the *centripetal* method reduces to the *chord-length* method, and, hence, the former can be considered as an extension to the latter. If  $a < 1$ , say  $a = 1/2$  (i.e., square root),  $|d_k - d_{k-1}|^a$  is less than  $|d_k - d_{k-1}|$ . Consequently, the impact of a longer chord (i.e.,  $length > 1$ ) on the length of the data points polyline is reduced, and the impact of a shorter chord (i.e.,  $length < 1$ ) on the length of the data polygon is increased. Because of this characteristic, the *centripetal* method handles sharp turns better than the *chord-length* method.

### 2.3.2 Generation of the knots vector

Once we have chosen the data parametrization method, we can proceed by generating the knots vector. Suppose we have  $s + 1$  parameters  $t_k$  with  $k = 0, \dots, s$  and degree  $p$ . For a B-spline curve of degree  $p$ , we need  $m + 1$  knots, where  $m = n + p + 1$ . If the curve is clamped, these knots are  $u_0 = u_1 = \dots = u_p = 0, u_{p+1}, \dots, u_{m-p-1}, u_{m-p} = u_{m-p+1} = \dots = u_m = 1$ . More precisely, the first  $p + 1$  and last  $p + 1$  knots are 0's and 1's, respectively. For the remaining  $n - p$  knots, they can be uniformly spaced or chosen properly to achieve some desired conditions. For B-spline fitting curves, the classical choice is to distribute them uniformly. The uniform knots distribution uniformly spaces the remaining  $n - p$  internal knots. Then,  $u_p = 0, u_{p+1}, \dots, u_{m-p-1}, u_{m-p} = 1$  divide  $[0, 1]$  into  $n - p + 1$  subintervals (or, knot spans). Therefore, the knots are

$$u_i = \begin{cases} 0 & 0 \leq i \leq p \\ \frac{i-p}{m-2p} & p < i < m - p \\ 1 & m - p \leq i \leq m \end{cases} \quad (2.16)$$

A uniformly spaced knots vector does not require the knowledge of the parameters, and is very simple to generate. Other techniques, as the *average* or *natural knots distribution*, can be applied to interpolating B-splines, not our case.

## Chapter 3

# RASH: The Reactive Affine Shaker

In the previous Chapter, we presented the B-spline curve model. A B-spline curve is completely determined by its knots vector and its control points. We have already discussed the problem of generating the knots vector, leaving to this Chapter the problem of determining the control point positioning. Control points stretch the B-spline curve toward them, acting as weights along the stripe connecting the first and the last data points, respectively  $d_0$  and  $d_s$ . Here, we assume to have fixed the number of control points to  $n + 1$ . The choice of the number of control points, and their initial assignment, is treated in Chapter 6. The control points positioning is commonly driven by an objective function for minimization. Such objective function measures the approximation error we want to minimize. The approximation error may be defined as the distance between the data points and their respective approximated values on the B-spline curve, determined through parametrization as previously discussed in Chapter 2. In Chapter 7, we discuss the new parametrization techniques proposals. Here, we assume to have already determined the parameters  $t_0, \dots, t_s$  that make the data points  $d_0, \dots, d_s$  to correspond to B-spline curve points  $C(t_0), \dots, C(t_s)$ . The goal of the objective function is to minimize the approximation errors  $|S(t_k) - d_k|$ ,

for  $k = 0, \dots, s$ . The classic approach for fitting B-splines considers the least squares formulation

$$E = \sum_{k=0}^s |S(t_k) - d_k|^2 \quad (3.1)$$

Standard numerical least squares methods include the Gauss-Newton algorithm (GNA), the gradient descent algorithm (GDA) and the Levenberg-Marquardt algorithm (LMA). In [10], the authors propose to use the LMA. They state that LMA tends to a quadratic convergence while the GDA approach shows a linear convergence rate only. In [12], the authors propose to optimize the spacing of the knots vector instead of control points, placing them along the fitting B-spline curve. They adopt the GDA method, stating that this approach simplifies the fitting problem by reducing the variables to optimize. The authors of [14] propose instead to consider the GNA, eliminating the parametrization problem in favor of approximation errors determined in respect to a local Frenet coordinate system. This technique is based on the approximated distance measure proposed in [13].

We propose instead to solve the least squares formulation of the fitting B-splines problem through the “Reactive Affine Shaker” (RASH) optimization algorithm. RASH is a reactive search optimization technique that doesn’t require to determine derivatives or gradients, resulting in a simpler and faster solution. The following presentation of RASH is based on [6].

### 3.1 The Reactive Affine Shaker Algorithm

The Reactive Affine Shaker algorithm (or RASH for short) is an adaptive random search algorithm based on function evaluations. The seminal idea of the scheme was presented for a specific application in neural computation in [3], which evolved proposing in RASH a more effective strategy during the initial part of the search by analyzing the evolution of the search direction in the first iterations, when the search succeeds with probability close to one.

### 3.1.1 Motivation and analysis

The algorithm starts by choosing at random, in the absence of prior knowledge, an initial point  $\mathbf{x}$  in the configuration space. This point is surrounded by an initial *search region*  $\mathcal{R}$  where the next point along the trajectory is searched for.

In order to keep a low computation overhead, the *search region* is identified by  $n$  vectors,  $\mathbf{b}_1, \dots, \mathbf{b}_n \in \mathfrak{R}^n$  which define a “box” around the point  $\mathbf{x}$ :

$$\mathcal{R} = \left\{ \mathbf{x} + \sum_{i=1}^n \alpha_i \mathbf{b}_i, \quad \alpha_1, \dots, \alpha_n \in [-1, 1] \right\}.$$

The search occurs by generating points in a stochastic manner, with a uniform probability in the search region. For a reason which will become clear in the following description, a single displacement  $\Delta$  is generated and two specular points  $\mathbf{x}^{(t)} + \Delta$  and  $\mathbf{x}^{(t)} - \Delta$  are considered in the region for evaluation (*double shot*, see also [7]). An evaluation is “successful” if the  $f$  value at at least one of the two trial points is better than the value at the current point.

By design, RASH is *an aggressive local minima searcher*: it aims at converging rapidly to the local minimizer in the attraction basin where the initial point falls.

The most computational effort during the search is spent by calculating function values  $f(\mathbf{x})$  at tentative points by assumption. Because of the algorithm simplicity, the assumption is valid for non-trivial real-world problems.

The search speed is related to the average size of the steps  $\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|$  executed along the search trajectory. Let’s consider two extreme cases. If the search region is very small and the function is smooth, the “double shot” strategy will produce a new successful point with probability close to one, but the step will be very small. *Vice versa*, if the search region is very large and it coincides with the initial range of interest, the search strategy will become that of naïve random search: points are generated at random in the entire search space. The step can be large, but the locality assumption is lost and,

unless the problem is very simple, a potentially very large number of points will have to be evaluated before finding a successful one. Ideally, to maximize the usage of the information derived from the costly  $f(\mathbf{x})$  computations, one should aim at the *largest possible step per function evaluation*. This optimal criterion cannot in general be fulfilled, in particular if the analytic form of the function is not known and values  $f(\mathbf{x})$  are obtained by simulation.

RASH aims at maintaining the search region size as large as possible, while still ensuring that the probability of a success per evaluation will be reasonably close to one. Success probabilities in the range 0.3 - 0.5 are considered acceptable. Now, the success probability is related both to the area of the search region, and to its form. For example, if the attractor basin consists of an elongated and narrow valley leading to a local minimizer, for a fixed area, a search region elongated along the bottom of the valley will guarantee a higher success rate of the double shot strategy, and therefore longer average step sizes.

RASH obtains both design objectives: (i) success probability per sample close to one and (ii) largest possible step size per successful sample, through a “reactive” determination of the search area during the search. For objective (i) the area is enlarged if the search is successful, reduced if unsuccessful, for objective (ii) the area is elongated along the last successful direction. Of course, “largest possible” has a heuristic meaning: given the partial knowledge about  $f$  and the lack of constraints about its functional form we are satisfied if a reasonably large step is determined by a simple reactive scheme.

With more detail, the algorithm proceeds by iterating the following steps:

1. A new tentative point is generated by sampling the local search region  $\mathcal{R}$  with a uniform probability distribution and by using the “double shot” strategy. The second specular shot is evaluated in a lazy manner only if the first one does not succeed.
2. The search region is modified according to outcome of the tentative point. It is compressed if the new function value is greater than the

current one (unsuccessful sample), it is expanded otherwise (successful sample). The region is modified by taking into account the direction of the last tentative step. In RASH, the search area defined by vectors  $\mathbf{b}_i$  undergoes an affine transformation.

3. If the sample is successful, the new point becomes the current point, and the search region  $\mathcal{R}$  is translated so that it becomes centered around the new point.

A last design decision concerns the initial size of the search area, in the absence of initial information about the local attraction basin of  $f$ . Two simple options, which do not require critical parameters to be tuned, are to start with a search area corresponding to the initial search range, which will be rapidly compressed in the following iterations until it leads to a success, or, on the contrary, to start with a very small search area, which will be rapidly expanded. The first option is in conflict with the requirement that RASH should scout for the local minimizer corresponding to the basin of attraction of the initial point. If arbitrarily large jumps are permitted at the beginning, all attraction basins could be reachable, with a probability depending on their sizes. Therefore RASH considers the second option.

When the function is smooth and the search region area goes to zero, the probability of success of the “double shot” strategy tends to one, no matter what the initial direction is. This fact creates an undesired effect: after picking the first tentative direction, one will have an uninterrupted sequence of successes. At each step, the search area will be expanded along the last direction, which in turn will be generated with uniform probability in an already elongated region. Through this self-reinforcing mechanism one may easily get an extremely elongated search region, where the elongation tends to be collinear with the first *random* direction, with no influence from the form of the current basin. To avoid this spurious effect, the expansion of the search region is isotropic in the initial part of the search, until the first unsuccessful direction is encountered, i.e., all box vectors are expanded by the same factor.

After explaining the design choices, let's now comment on the name (Reactive Affine Shaker). The solver's movements try to minimize the number of jumps towards the minimum region, and this is achieved by constantly changing the movement direction and size. Search region and therefore step adjustments are implemented by a feedback loop guided by the evolution of the search itself, therefore implementing a "reactive" self-tuning mechanism. The constant change in step size and direction creates a "shaky" trajectory, with abrupt leaps and turns. Last, modifications of the search parameters are through an affine transformation on the shape of the search region.

### 3.1.2 RASH pseudo-code

Details of the RASH algorithm are shown in Figure 3.1. At every iteration, a displacement  $\Delta$  is generated so that the point  $\mathbf{x} + \Delta$  is uniformly distributed in the local search region  $\mathcal{R}$  (line 4). To this end, the basis vectors are multiplied by different random numbers in the real range  $[-1, 1]$  and added:

$$\Delta = \sum_j \text{Rand}(-1, 1) \mathbf{b}_j.$$

$\text{Rand}(-1, 1)$  represents a call of the random-number generator. If one of the two points  $\mathbf{x} + \Delta$  or  $\mathbf{x} - \Delta$  improves the function value, then it is chosen as the next point. Let us call  $\mathbf{x}'$  the improving point. In order to enlarge the box along the promising direction, the box vectors  $\mathbf{b}_i$  are modified as follows. The direction of improvement is  $\Delta$ . Let us call  $\Delta'$  the corresponding vector normalized to unit length:

$$\Delta' = \frac{\Delta}{\|\Delta\|}.$$

Then the projection of vector  $\mathbf{b}_i$  along the direction of  $\Delta$  is

$$\mathbf{b}_i|_{\Delta} = \Delta'(\Delta' \cdot \mathbf{b}_i) = \Delta' \Delta'^T \mathbf{b}_i.$$

To obtain the desired effect, this component is enlarged by a coefficient  $\rho > 1$ ,

Variable	Scope	Meaning
$f$	(input)	Function to minimize
$\mathbf{x}$	(input)	Initial point
$\mathbf{b}_1, \dots, \mathbf{b}_d$	(input)	Vectors defining search region $\mathcal{R}$ around $\mathbf{x}$
$\rho$	(input)	Box expansion factor
$t$	(internal)	Iteration counter
$\mathbf{P}$	(internal)	Transformation matrix
$\mathbf{x}, \Delta$	(internal)	Current position, current displacement

1. **function** AffineShaker ( $f, \mathbf{x}, (\mathbf{b}_j), \rho$ )

```

2.    $t \leftarrow 0$ ;
3.   repeat
4.      $\Delta \leftarrow \sum_j \text{Rand}(-1, 1)\mathbf{b}_j$ ;
5.     if  $f(\mathbf{x} + \Delta) < f(\mathbf{x})$ 
6.        $\mathbf{x} \leftarrow \mathbf{x} + \Delta$ ;
7.        $\mathbf{P} \leftarrow \mathbf{I} + (\rho - 1) \frac{\Delta\Delta^T}{\|\Delta\|^2}$ ;
8.     else if  $f(\mathbf{x} - \Delta) < f(\mathbf{x})$ 
9.        $\mathbf{x} \leftarrow \mathbf{x} - \Delta$ ;
10.       $\mathbf{P} \leftarrow \mathbf{I} + (\rho - 1) \frac{\Delta\Delta^T}{\|\Delta\|^2}$ ;
11.     else
12.        $\mathbf{P} \leftarrow \mathbf{I} + (\rho^{-1} - 1) \frac{\Delta\Delta^T}{\|\Delta\|^2}$ ;
13.      $\forall j \mathbf{b}_j \leftarrow \mathbf{P} \mathbf{b}_j$ ;
14.    $t \leftarrow t+1$ 
15. until convergence criterion;
16. return  $\mathbf{x}$ ;

```

Figure 3.1: The Affine Shaker algorithm

so the expression for the new vector  $\mathbf{b}'_i$  is

$$\begin{aligned}
\mathbf{b}'_i &= \mathbf{b}_i + (\rho - 1)\mathbf{b}_i|_{\Delta} & (3.2) \\
&= \mathbf{b}_i + (\rho - 1)\Delta'\Delta'^T\mathbf{b}_i \\
&= \mathbf{b}_i + (\rho - 1)\frac{\Delta\Delta^T}{\|\Delta\|^2}\mathbf{b}_i \\
&= P\mathbf{b}_i
\end{aligned}$$

where

$$P = \mathbf{I} + (\rho - 1)\frac{\Delta\Delta^T}{\|\Delta\|^2}. \quad (3.3)$$

The fact of testing the function improvement on both  $\mathbf{x} + \Delta$  and  $\mathbf{x} - \Delta$  is called *double-shot strategy*: if the first sample  $\mathbf{x} + \Delta$  is not successful, the specular point  $\mathbf{x} - \Delta$  is considered. This choice drastically reduces the probability of generating two consecutive unsuccessful samples. The motivation is clear if one considers differentiable functions and small displacements: in this case the directional derivative along the displacement is proportional to the scalar product between displacement and gradient  $\Delta \cdot \nabla f$ . If the first is positive, a change of sign will trivially cause a negative value, and therefore a decrease in  $f$  for a sufficiently small step size. The empirical validity for general functions, not necessarily differentiable, is caused by the correlations and structure contained in most of the functions corresponding to real-world problems.

If the double-shot strategy fails, then the affine transformation (3.2) is applied by replacing the expansion factor  $\rho$  with its inverse  $\rho^{-1}$  (line 12 of Figure 3.1), causing a compression of the search area.

An illustration of the geometry of the Reactive Affine Shaker algorithm is shown in Figure 3.2, where the function to be minimized (in this case the domain is a square in  $n = 2$  dimensions) is represented by a contour plot showing *isolines* at fixed values of  $f$ , and two trajectories (ABC and A'B'C') are plotted. The search regions are shown for some points along the search trajectory. A couple of independent vectors define the search region as a parallelogram centered on the point. The design criteria are given by an

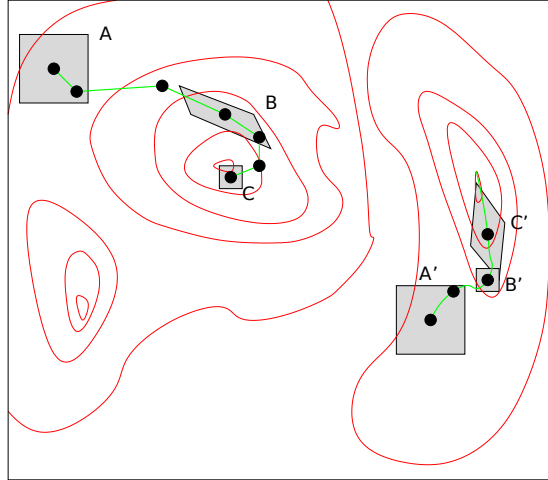


Figure 3.2: Affine Shaker geometry: two search trajectories leading to two different local minima. The evolution of the search regions is also illustrated

*aggressive search for local minima*: the search speed is increased when steps are successful (points A and A' in Figure 3.2), reduced only if no better point is found after the double shot. When a point is close to a local minimum, the repeated reduction of the search frame produces a very fast convergence of the search (point C in Figure 3.2). Note that another cause of reduction for the search region can be a narrow descent path (a “canyon”, such as in point B' of Figure 3.2), where only a small subset of all possible directions improves the function value. However, once an improvement is found, the search region grows in the promising direction, causing a faster movement along that direction.

### 3.1.3 Termination and repeated runs

For most continuous optimization problems, an effective estimation of the number of steps required for identifying a global minimum is clearly impossible. Even when a local minimum is located, it is generally impossible to

determine whether it is the global one or not, in particular if the knowledge about the function derives only from evaluations of  $f(x)$  at selected points.

Because RASH does not include mechanisms to escape from local minima, it should be stopped as soon as the trajectory is sufficiently close to a local minimizer. Suitable termination criteria can be derived, for instance a single RASH run can be terminated if the search region becomes smaller than a threshold value. In fact, the box tends to reduce its volume in proximity of a local minimum because of repeated failures in improving the function value.

# Chapter 4

## B-spline fitting with RASH

We discuss here our approach for reconstructing a curve given only by an ordered sequence of  $s + 1$  sparse and noisy data points  $d_0, \dots, d_s$ . We consider B-splines as solution models. We want the fitting curve to be continuous until the second derivative and to pass through the first and the last data points, respectively  $d_0$  and  $d_s$ . We assume that for all data points  $d_k = (x_k, y_k)$ ,  $x_k \in [0, 1]$  for simplicity. One does not lose generality because B-spline curves have the affine invariance property, so any translation and scaling operation can be easily reversed. Given the data points  $d_0, \dots, d_s$ , we have to determine their B-spline fitting curve, which is completely determined through its knots vector and its control points. The control points position is determined through an iterative optimization process. The optimizer is based on the RASH algorithm [5], presented in the previous Chapter. In this Chapter, we discuss more precisely what we consider the optimal solution and how we measure the distance to it, defining accordingly the objective function to minimize. The choice of the initial solution and the adopted parametrization scheme are crucial to improve the optimization performances. They will be discussed separately in the next Chapters. Figure 4.1 shows an example of B-spline curve that fits a set of data points. Data points are represented by circles, while control points are represented by triangles. The minimum distances between data points and the B-spline curve are represented by arrows.

These distances measure the approximation error. In the next section, we try to characterize in details the optimal solution.

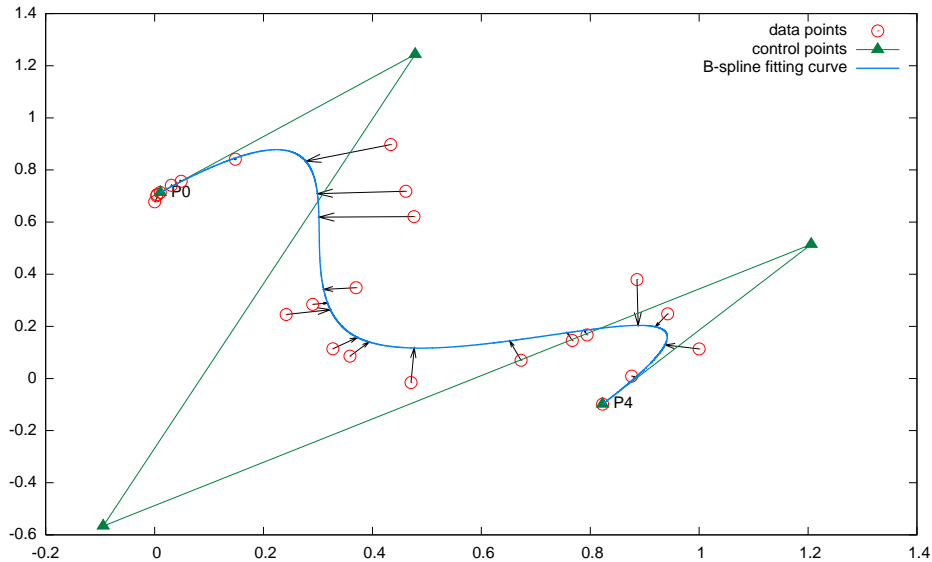


Figure 4.1: The problem of fitting with a B-spline a set of data points

## 4.1 Characterization of the optimal solution

Our objective is to determine a B-spline fitting curve  $C$  such that it follows closely the given  $s + 1$  data points  $d_0, \dots, d_s$ , taking care of their noise and sparsity. This curve shouldn't memorize the data points, but instead it should appropriately generalize. This property can be achieved by choosing the proper number of control points, a measure of how much information a B-spline curve can memorize. As we said, we are interested in memorizing only the bare minimum to model properly the noisy data points, without memorizing them exactly. A simpler B-spline fitting curve is desirable not just for practical convenience, but also because it allows us to determine a compact model for the data points. The choice of the number of control points, and their initial positioning, are discussed separately in Chapter 6. Once the number of control points is chosen, we need to position them prop-

erly. Their optimal positioning is such that it minimizes the approximation error for each data point. The distance of the data points  $d_0, \dots, d_s$  to the B-spline fitting curve  $C$  should be minimal. From those considerations, we proceed in the next section to propose a simple "goodness of fit" measure, which measures how well our model fits the data points.

## 4.2 A simple "goodness of fit" measure

In our case, we are interested in a measure which is invariant to rotation. Therefore, given an experimental data point we seek for the minimum distance to the curve. We define the energy function  $E_{gof}$  we want to minimize as follows:

$$E_{gof} = \sum_{k=0}^s \frac{|C(t_k) - d_k|^2}{s + 1} \quad (4.1)$$

Where  $t_k$  are the parameters which lead to the minimum distance between the  $C$  curve and the data points  $d_0, \dots, d_s$ .

The term "goodness of fit" derives from statistics, where the goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question. Such measures can be used in statistical hypothesis testing.

This measure has been used as evaluation function for the preliminary experiments presented in the next Chapters. Determining an approximation of this measure requires to sample the B-spline fitting curve  $C$ , we discuss in Chapter 5. A more efficient measure which serves as objective function for the optimization process is introduced in the next section.

## 4.3 Objective function for minimization

In this section, we define more precisely the function to minimize during the optimization process. The classical parametrization schemes we presented in

Chapter 2 and the new proposals we will discuss in Chapter 7 permit to define a more efficient measure of the distance between the B-spline fitting curve  $C$  and the data points  $d_0, \dots, d_s$ . Let's recall that through parametrization, we pick  $s + 1$  parameters  $t_k, k = 0, \dots, s$  so that point  $C(t_k)$  correspond to data point  $d_k$ . The approximation error for data point  $d_k$  can be measured as the Euclidean distance between  $d_k$  and its corresponding point on the B-spline curve  $C$  determined through its respective parameter  $t_k, C(t_k): |C(t_k) - d_k|$ . In general, there is no guarantee that  $C(t_k)$  is the closest point to  $d_k$ , it is only an approximation which can be determined more efficiently. In the experimental evaluation, we will compare different parametrization schemes, discussing their influence.

Similarly to  $E_{gof}$ , we measure the distance to the optimal solution with the Root Mean Square (RMS) error of  $|C(t_k) - d_k|$ , for  $k = 0, \dots, s$ . The RMS error is the square root of the average of the square of the error introduced by each approximated data point  $C(t_k)$

$$RMS = \sqrt{\frac{\sum_{k=0}^s |C(t_k) - d_k|^2}{s + 1}} \quad (4.2)$$

Because we are minimizing this value, we can avoid the square root, also for a better computation efficiency. And therefore, the energy function that we are using is

$$E_{param} = \frac{\sum_{k=0}^s |C(t_k) - d_k|^2}{s + 1} \quad (4.3)$$

Note that the  $E_{param}$  measure depends on the adopted parametrization technique, so we cannot compare different parametrization techniques comparing directly their respective  $E_{param}$  values. The  $E_{param}$  measure is an approximation of  $E_{gof}$  which we use as objective function for minimization, but that we cannot use for evaluations or comparisons. We will discuss in details how to evaluate and compare different techniques in Chapter 5. Once the objective function for minimization is defined, we can proceed discussing the optimization process.

## 4.4 The optimization process

In the previous section, we have defined the objective function for minimization to be considered in our optimization process. The RASH optimization algorithm, previously presented in Chapter 3, requires in input an objective function, a set of variables to optimize and a set of intervals of their admissible values. Let's recall that a B-spline curve is completely determined by its knots vector  $u_0, \dots, u_m$ , its control points  $P_0, \dots, P_n$  and the degree  $p$  of the basis functions, which we have fixed to three. We consider as variables of the optimization process the control points of the B-spline curve  $C$ , each composed by two coordinates. The first and the last control points are fixed, so that we have just  $2(n - 1)$  scalar variables to optimize. The intervals of admissible values for these variables depend on the curves we want to fit, there isn't in general an upper and lower limit. We will discuss how to adaptively choose the number of control points and their respective intervals of admissible values in Chapter 6. Figure 4.2 reports an example of fitting B-spline determined with our approach.

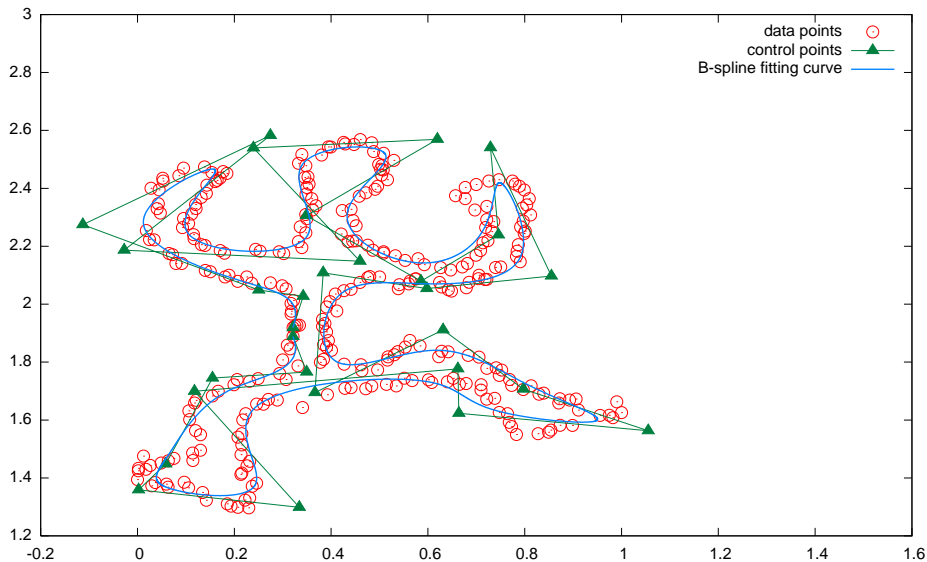


Figure 4.2: B-spline approximated curve determined through the optimization process

## Chapter 5

# Evaluation of the optimization process

In order to compare and evaluate different techniques, we need an evaluation error measure independent of the objective function and the optimization process. The computation of this measure doesn't influence the speed of the optimization process, so it may require much more time to be determined than the RMS approximation error measure  $E_{param}$  considered in the objective function. We need also a method for generating controlled sparse and noisy data points. We can consider as evaluation error measure an approximation of  $E_{gof}$ , determined by sampling the B-spline fitting curve  $C$  and considering as approximation error for each data point the minimum distance between the data point and the set of samples of the  $C$  curve. These samples of the  $C$  curve can be determined stepping from zero to one by increments of  $\delta$ , with  $\delta$  reasonably small to sufficiently capture the shape of the  $C$  curve. In the next section, we address the problem of constructing the test curve models, introducing sparsity and noise in their derived data points.

## 5.1 Generation of test curves

In order to stress our proposed approach and compare the various possible choices, we consider here the problem of constructing a series of noisy and sparse data points  $d_k$ , with  $k = 0, \dots, s$  representing the curve  $D$  on the domain interval  $[a, b]$ . The curve  $D$  can be a line, a sine wave or a spline curve interpolating a set of given data points. These curves can describe sufficiently well trajectories of moving objects or other simple phenomena. We proceed by describing how to sparsely sample the curve  $D$ , introducing also the desired noise in a second step. The sparsity of data points is modeled by picking  $s + 1$  samples from the uniform distribution in the interval  $[a, b]$ , reordered in ascending order obtaining the sequence  $U_0, \dots, U_s$  and a sparse sampling of the curve  $D$ :  $D(U_0), \dots, D(U_s)$ . The next step is to add noise. The noise of the data points is modeled with a standard normal distribution through the Box-Muller transform [4], a method of generating pairs of independent standard normally distributed (expectation 0, variance 1) random numbers. Given two samples  $X_0, X_1$  from the uniform distribution on the interval  $(0, 1)$ , the Box-Muller transform maps them to two normally distributed samples:

$$Z_0 = \sqrt{-2\ln X_0} \cos(2\pi X_1) \quad (5.1)$$

$$Z_1 = \sqrt{-2\ln X_0} \sin(2\pi X_1) \quad (5.2)$$

Approximately the 99.7% of the values of the standard normal distribution are in the interval  $[-3, 3]$ , so the 99.7% of the generated values  $(Z_0, Z_1)$  stay in a circumference of radius 3 centered at the origin,  $(0, 0)$ . We can define the sequence of sparse noisy data points  $d_k$ , with  $k = 0, \dots, s$  as follows:

$$d_k = (U_k + \frac{\alpha}{3}Z_0, D(U_k) + \frac{\alpha}{3}Z_1) \quad (5.3)$$

The value of  $\alpha$  determines how much noise we want to introduce and it is a parameter of the test curve generator. With  $\alpha = 0$  there isn't any noise,

with  $\alpha = 1$  the 99.7% of the noise is normally distributed in the range  $[-1, 1]$ . Finally, for simplicity, we translate and scale the data points  $d_k$  from the interval  $[a, b]$  to the interval  $[0, 1]$  through simple translation and scaling operations in the Euclidean space. We report now an example taking as  $D$  curve a sinusoid in the range  $[0, 6\pi]$  with amplitude 1, angular frequency 1 and phase 0. In Figure 5.1  $\alpha = 0$  while in 5.2  $\alpha = 0.5$ . Increasing the value of  $\alpha$ , we increase also the amount of noise as expected.

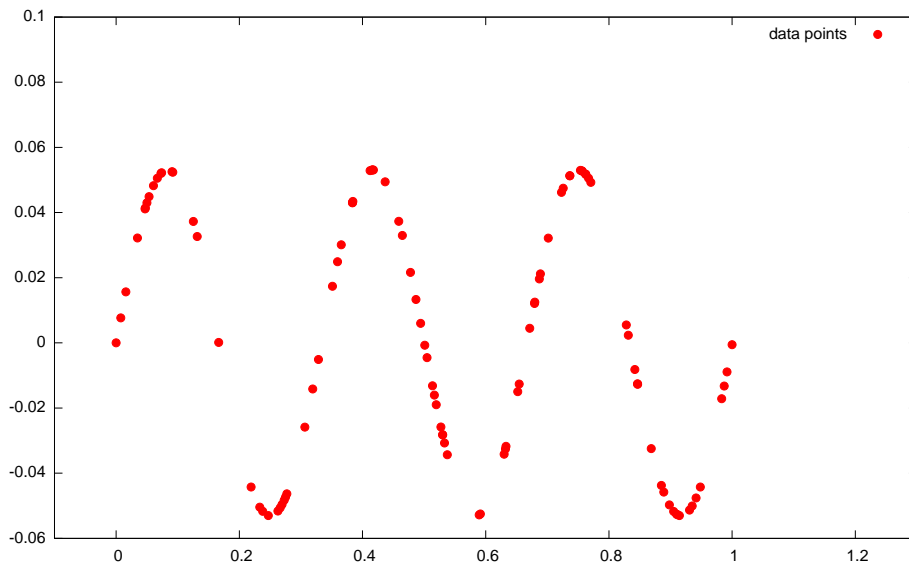


Figure 5.1: Data points derived sampling the sinusoid  $\sin(x)$  in interval  $[0, 6\pi]$  with  $\alpha = 0$

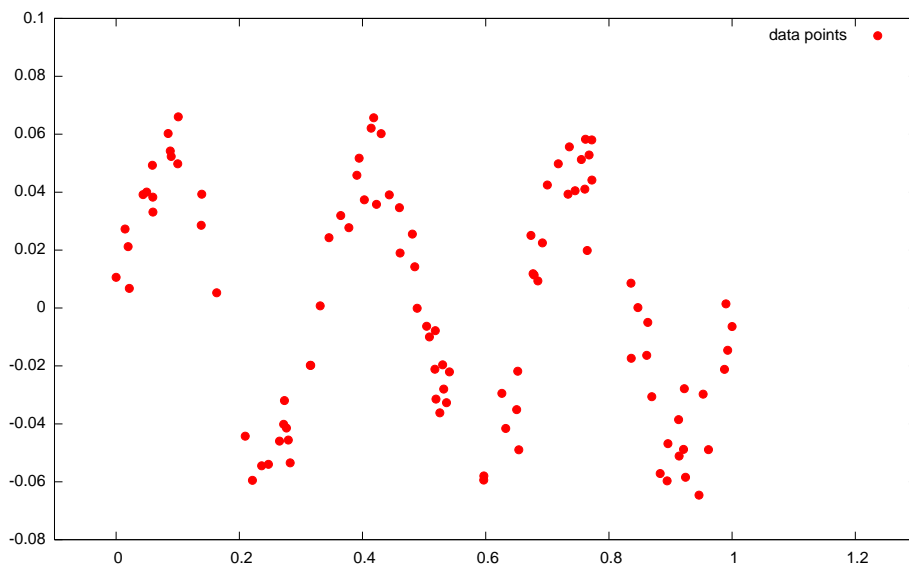


Figure 5.2: Data points derived sampling the sinusoid  $\sin(x)$  in interval  $[0, 6\pi]$  with  $\alpha = 0.5$

## Chapter 6

# Influence of the initial solution and number of control points

The choice of the initial solution may impact significantly on the performances of the optimizer [18]. A good initial solution can speed up significantly the optimization process, as we will demonstrate in this Chapter. We propose and discuss some possible approaches to this problem with evaluations and comparisons. Let's recall that, in order to completely construct an initial B-spline fitting curve  $C$ , we must determine the knots vector  $u_0, \dots, u_m$  and the control points  $P_0, \dots, P_n$ . We have that  $m$  and  $n$  must satisfy the equality  $m = n + p + 1$ , so once the number of control points is fixed, the number of knots is also fixed. The first choice we must face is how many control points we want our solutions to have. Once  $n$  is chosen, we have fixed the number of control points to  $n + 1$  and the number of knots to  $n + p + 1$ . After this step, we can proceed by initializing properly the knots vector.

A technical issue regards the problem of fixing the extreme points of the B-spline curve. In detail, we want the B-spline fitting curve to pass exactly through the first and last data points, so we have to set the first  $p + 1$  and last  $p + 1$  knots to 0's and 1's, respectively. We set the remaining internal  $n - p$  knots by considering the uniform knots distribution which has been discussed in the B-spline introduction in Chapter 2. The knots vector

$u_0 = u_1 = \dots = u_p = 0, u_p + 1, \dots, u_{m-p-1}, u_{m-p} = u_{m-p+1} = \dots = u_m = 1$  is now initialized. The clamped knots vector requires also to initialize the first and the last control points respectively to the first and last data points:  $P_0 = d_0$  and  $P_n = d_s$ . The problem is now reduced to the choice of the  $n - 1$  internal control points  $P_1, \dots, P_{n-1}$ .

As a first attempt to define them, we can identify the region of their admissible values. This region unfortunately depends on the particular phenomena the data points are representing, and there isn't a general solution which always works. This limitation, which is part of the optimizer initialization and not of the initial solution, requires at least an analysis of the data points, and can be determined during the control points initialization. For our experimental curves, an interval of  $[-10, 10]$  on the two axes is empirically sufficient, and we will consider it as the largest possible interval for each control point. This interval is sufficient because the test curves have limited curvature and can be modeled by control points in the given intervals. In the next sections, we will discuss different strategies to initialize the internal control points. The questions we want to answer are: Which initialization techniques speed up the optimization process? How robust is the optimization process? Can we determine adaptively the number of control points? The rest of this Chapter tries to answer these questions.

## 6.1 Number of control points

We discuss here about the inferior and superior limits of the number of control points. Let us start from the lower bound. With a clamped knots vector, we can also define more precisely how to choose the number of control points. Let's recall that with a clamped knots vector, we have that the first and last  $p + 1$  knots must be initialized properly as previously discussed. We need at least  $2(p + 1)$  knots. We have also that  $m = n + p + 1$ , so we have that

$$n + p + 1 \geq 2(p + 1) \tag{6.1}$$

If we fix the degree of the basis functions  $p$  to three, we have that  $n \geq 4$ . The number of control points is equal to  $n + 1$  by definition, so we need at least five control points in order to define a B-spline fitting curve with basis functions of degree three and with a clamped knots vector, which ensures the B-spline fitting curve to pass exactly through the first and last data points. This is a lower bound for the number of control points.

The upper bound can be defined as the number of data points  $s + 1$ : Obviously, we won't consider a number of control points greater than the number of data points, because this always leads to overfitting. We have that  $n$  must be a value chosen in the range  $[4, s]$ . We assume to have at least five data points for simplicity.

The number of control points of a B-spline is a measure of the flexibility and complexity of the model, so its choice must be driven by the complexity of the curve we want to fit: We should consider only the complexity of the curve, and not the noise of the derived data points. This facilitates the resulting B-spline to model such curve in spite of the sparsity and noise.

Figure 6.1 shows a B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 4\pi]$ , reconstructed from 100 sparse and noisy data points, after 2000 optimization steps. We have six control points, the required amount of model flexibility we need in order to fit the curve without overfitting. Note that the control points are positioned in the regions of higher curvature.

## 6.2 Random control points initialization

The most direct and simple approach is to initialize the  $n - 1$  internal control points to completely random values. The number of control points, without any analysis or knowledge of the process which is generating the data points, is a parameter of the system that must be chosen empirically. We call this method the *random* initialization scheme.

We present in Figure 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 and 6.8 the resulting B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200

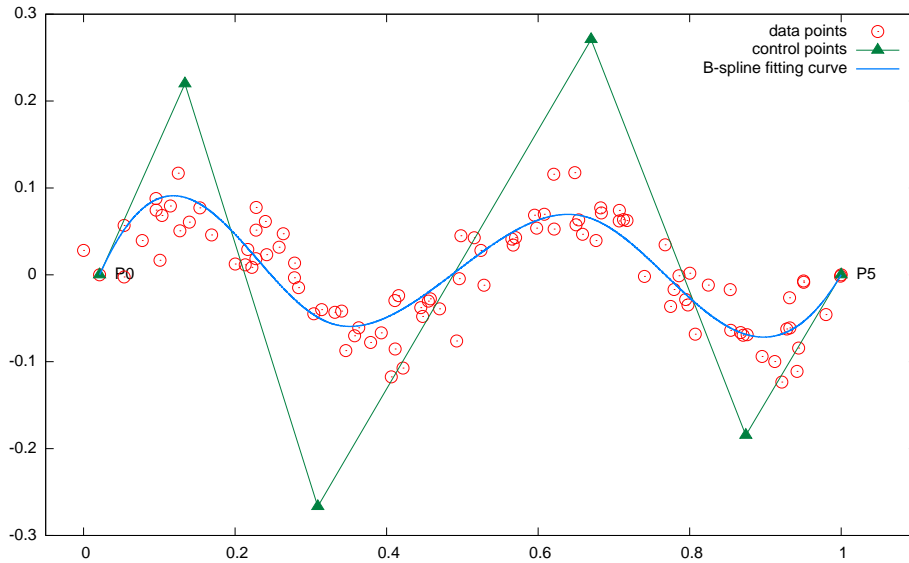


Figure 6.1: B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 4\pi]$ , derived from 100 sparse and noisy data points, after 2000 optimization steps

sparse and noisy data points, with noise level of  $\alpha = 0.5$  after respectively 0, 200, 400, 600, 800, 1000 and 2000 optimization steps. The values of  $E_{gof}$  and  $E_{param}$  have been also reported in Figure 6.9. For this example, we fixed  $n = 9$ , so we have 10 control points. Note that  $E_{param}$  values may be initially bigger than  $E_{gof}$  values, because the  $E_{gof}$  measure considers the minimum distances between the data points and the B-spline fitting curve. In this case, the minimum distances are always less than or equal to the distances between the data points and the B-spline curve extremes,  $C(0) = d_0, C(1) = d_s$ . This doesn't happen for  $E_{param}$ , which depends on the adopted parametrization technique.

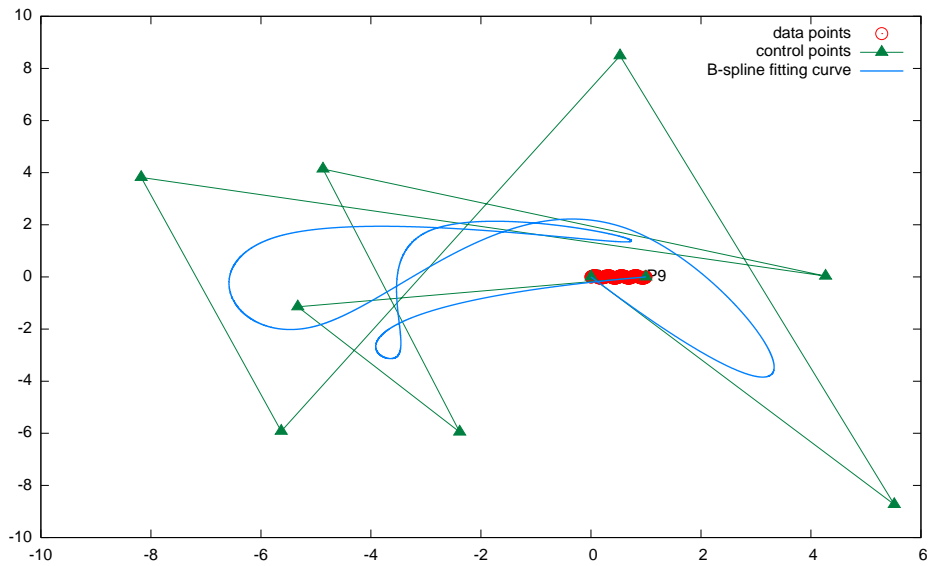


Figure 6.2: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 0 optimization steps

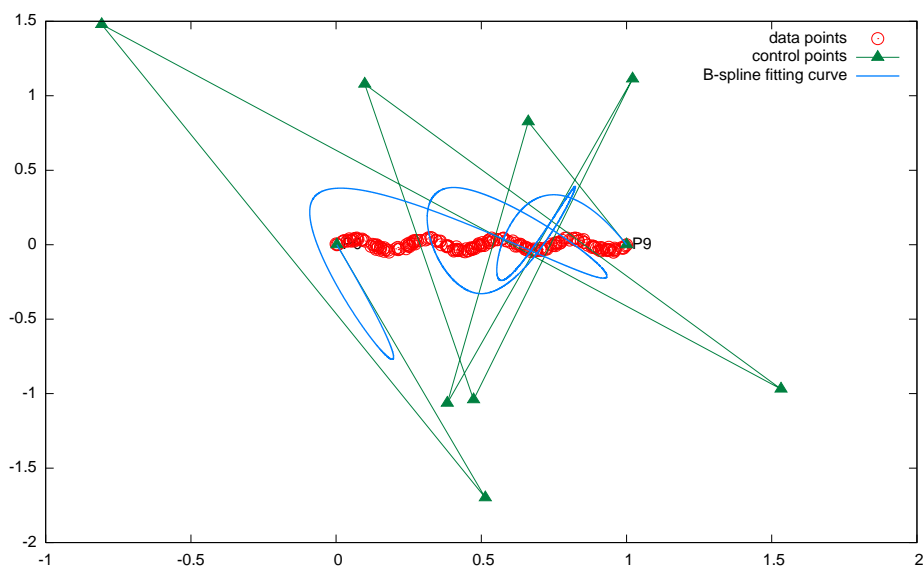


Figure 6.3: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 200 optimization steps

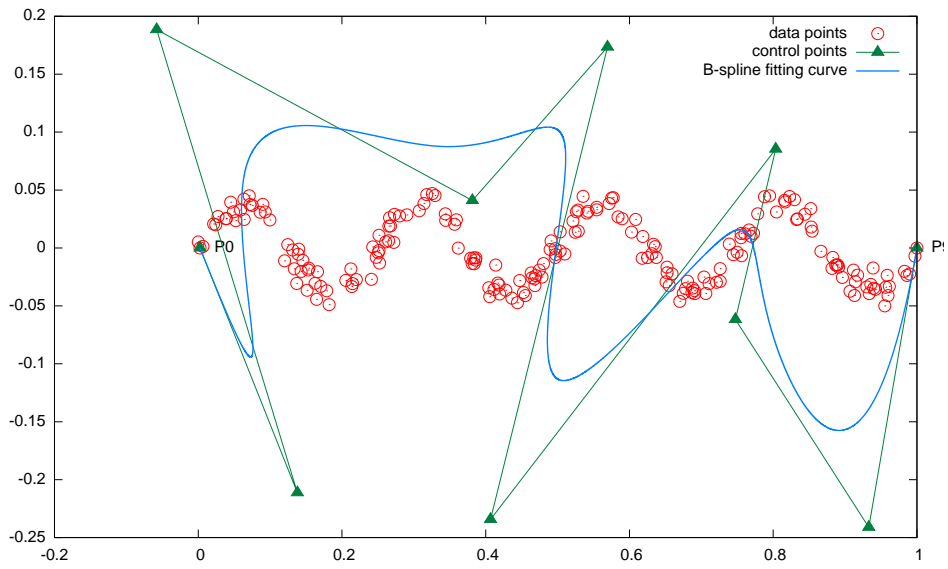


Figure 6.4: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 400 optimization steps

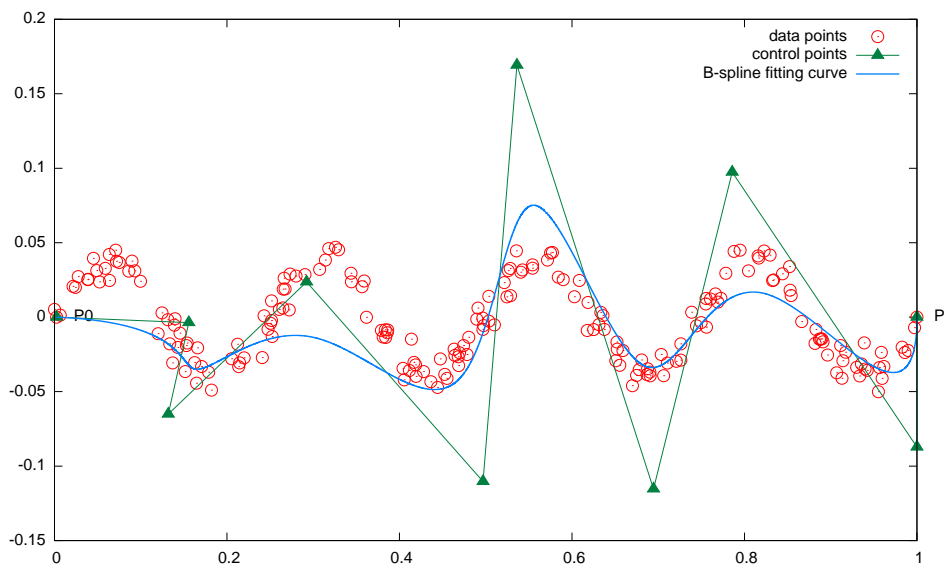


Figure 6.5: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 600 optimization steps

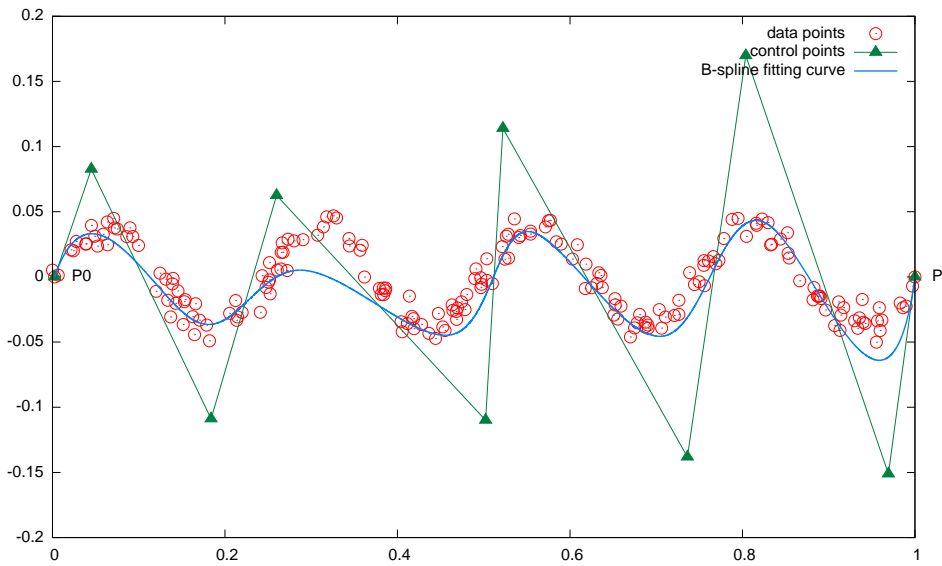


Figure 6.6: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 800 optimization steps

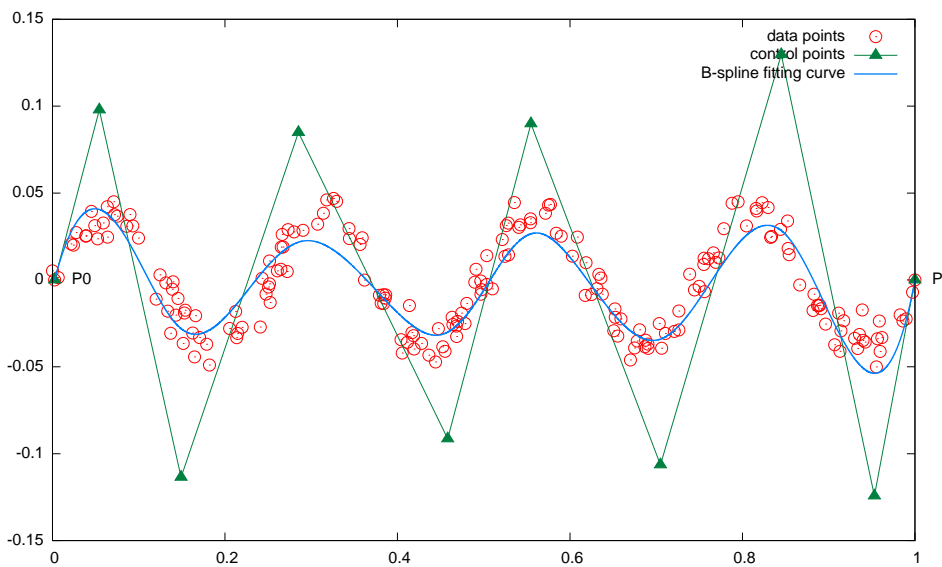


Figure 6.7: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 1000 optimization steps

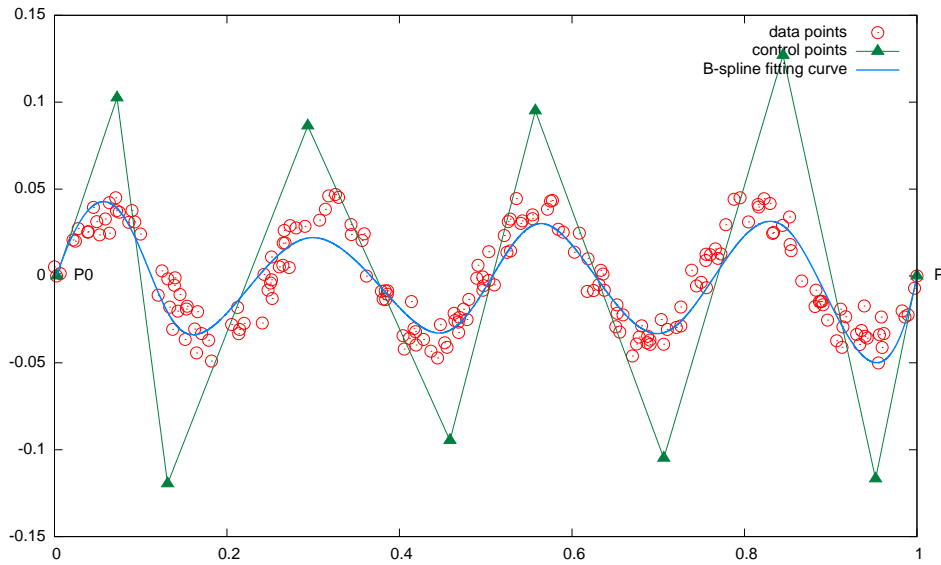


Figure 6.8: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  after 2000 optimization steps

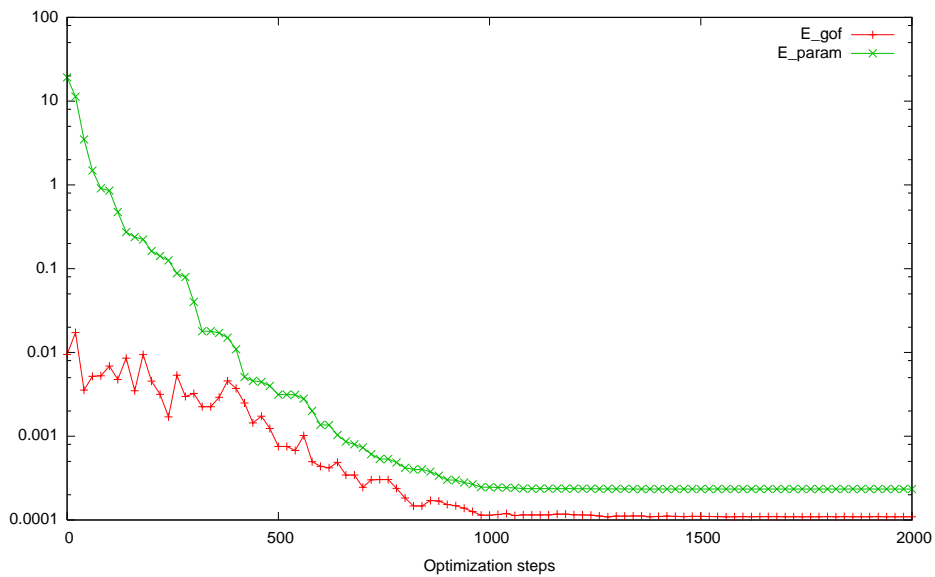


Figure 6.9: Comparison of  $E_{gof}$  and  $E_{param}$  for B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$ , 2000 optimization steps, *random* control points initialization

### 6.3 Random subset of data points as control points initialization

We can improve the *random* initial solution by picking as internal control points a sparse subset of the data points  $d_0, \dots, d_s$ , proceeding in two steps: In the first step, we pick a subset of  $n - 1$  random data points, in the second step we reorder them as they appear in the data points sequence  $d_0, \dots, d_s$ . We take them as internal control points. The resulting B-spline fitting curve will pass "near" the desired curve and the optimizer will benefit from this. The number of control points to consider is a parameter of the system that must be chosen empirically. We call this method the *random subset* initialization scheme.

We present in Figure 6.10 the initial B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ . For this example, we considered  $n = 9$ , so we have 10 control points as in the previous example. We observe that the shape of the desired curve is at least partially similar, so it should be also a better initial solution. Our expectation is confirmed by the comparison of the *random* control points and the *random subset* initialization schemes, reported in Figure 6.11.

### 6.4 Uniform control points initialization

For some particular curves, such as lines and sinusoids, another approach that may work successfully consists of initializing control points uniformly along the line connecting the first and last data points, respectively  $d_0$  and  $d_s$ .

We present in Figure 6.12 the initial B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ . For this example, we considered  $n = 9$  as in the last examples. In Figure 6.13, we report the comparison between the *random sub-*

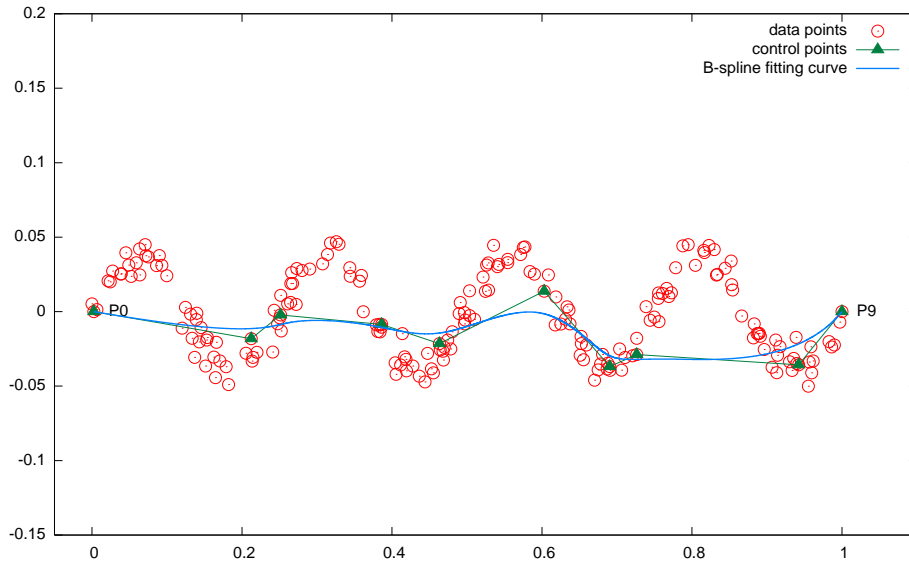


Figure 6.10: Initial solution determined picking a *random subset* of data points as initial control points

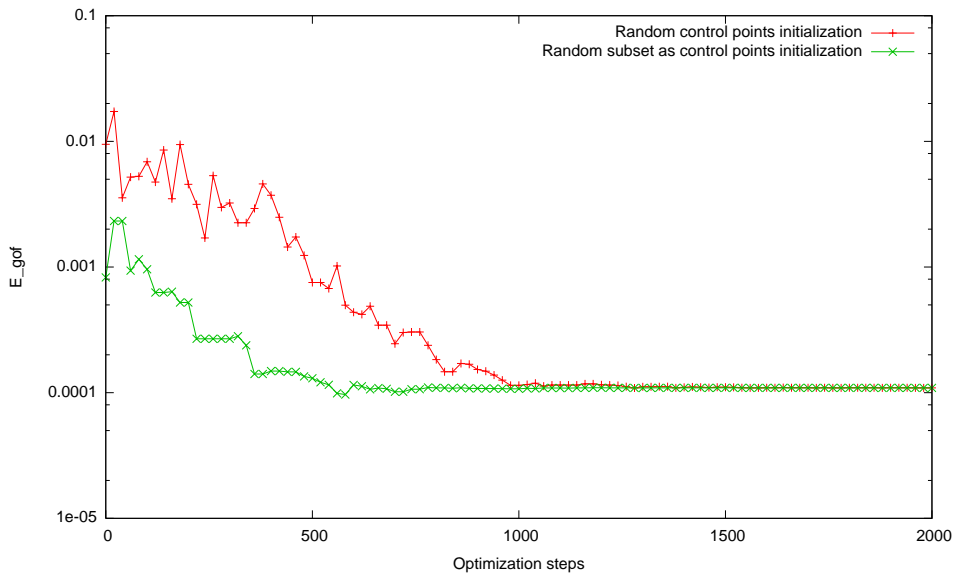


Figure 6.11: Comparison between *random* and *random subset* initialization schemes of  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse and noisy data points

*set* initialization scheme and the *uniform* initialization scheme. The uniform placement seems more robust, but the convergence is slower. In general, this approach works only for simple curves like lines, sinusoids or splines, but it won't work with more complex shapes as parametric curves because a line as initial solution, as in this case, may be totally wrong. The *random subset* initialization scheme remains the best choice.

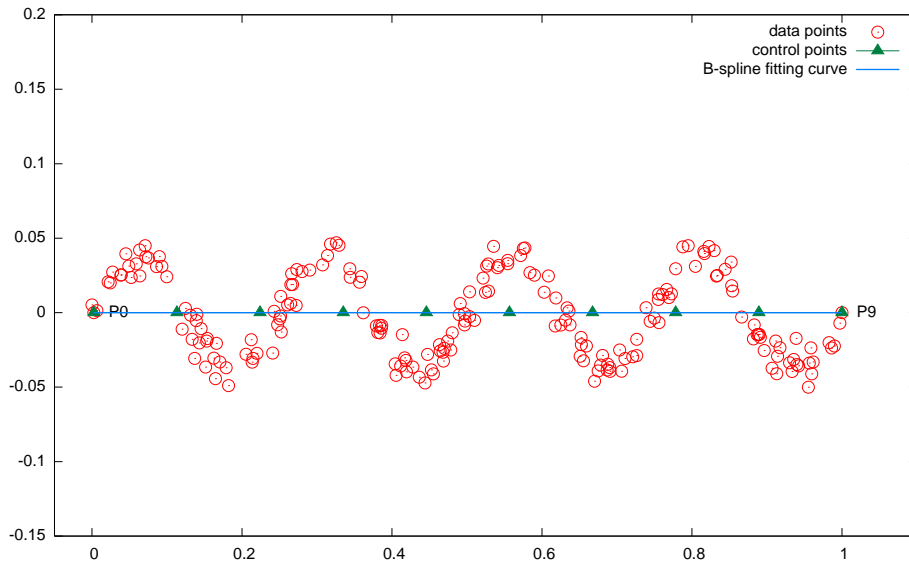


Figure 6.12: Initial solution determined through uniformly spaced control points along the segment connecting the two extreme data points

## 6.5 Reactive choice of number and position of control points

In the previous sections, we have proposed simple initialization schemes that don't take the shape of the curve we want to fit into account, positioning the control points randomly or near the curve region. The initial positioning of the control points can speed up the convergence toward a local minimum, while the choice of the number of control points impacts significantly

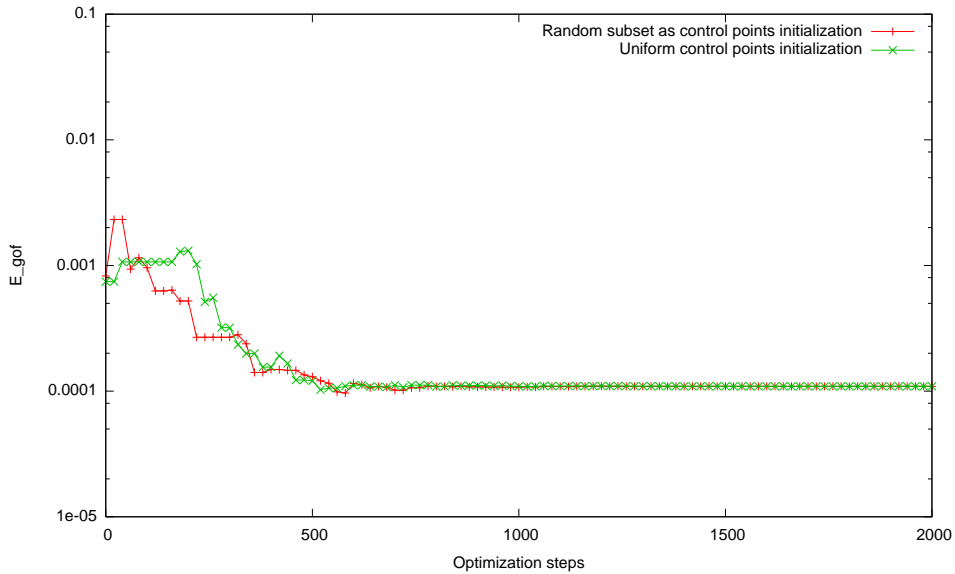


Figure 6.13: Comparison between *random subset* and *uniform* initialization schemes of  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse and noisy data points

the quality of the determined solution. We have tried to address the first problem, leaving the second to an empirical choice. We consider now both problems: The first refers in particular to the initial value of the optimization process variables, while the second refers to an adaptive choice of the number of variables to optimize. Those two problems are related, because both depend on the complexity of the curve and on the noisy process generating the experimental points. A first intuition suggests to consider the direction change of consecutive data points. In the next sections, we propose new techniques that can determine adaptively the number and the initial positioning of the control points, in presence of noise.

### 6.5.1 Slope-based initial solution

We can try to improve further the initial assignments of control points by analyzing the data points more deeply. Let's recall that we want to specify an initial B-spline fitting curve such that it coarsely approximates the shape of

the target curve. Control points of B-spline curves are used as weights of the basis functions, stretching the curve toward their position. Their placement should be near the regions of the curve which have a high curvature. If we distribute control points in these regions, we can approximatively capture the shape of the curve we want to fit. The first step is to recognize these particular regions. A first possible approach consists of measuring the curve direction change through the slope change of the segments identified by any two consecutive data points. The slope of a segment can be measured with the angular coefficient of the line passing through its two extremes. We proceed by considering as points inside those high curvature zones any data point  $d_k$  such that the slope  $m_{k-1}$  of the segment identified by data points  $d_{k-1} = (x_{k-1}, y_{k-1}), d_k = (x_k, y_k)$  is significantly different from the slope  $m_k$  of the segment identified by data points  $d_k = (x_k, y_k), d_{k+1} = (x_{k+1}, y_{k+1})$ , where the slopes are defined as follows:

$$m_{k-1} = \frac{y_{k-1} - y_k}{x_{k-1} - x_k} \quad (6.2)$$

$$m_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \quad (6.3)$$

We consider as relevant change between the angular coefficients  $m_{k-1}, m_k$  any of the following conditions:

$$(m_{k-1} < 0 \wedge m_k > 0) \vee (m_{k-1} > 0 \wedge m_k < 0) \quad (6.4)$$

$$(m_{k-1} < 1 \wedge m_k > 1) \vee (m_{k-1} > 1 \wedge m_k < 1) \quad (6.5)$$

$$(m_{k-1} > -1 \wedge m_k < -1) \vee (m_{k-1} < -1 \wedge m_k > -1) \quad (6.6)$$

At least one condition holds if the curve passes through any of the following lines, which together divide the Euclidean space in eight equal regions

meeting at the origin,  $(0, 0)$ :

$$y = 0, x = 0, y = x, y = -x \tag{6.7}$$

The complete algorithm is presented in Algorithm 1.

In Figure 6.14, we report the resulting initial solution for a set of data points representing a partial circle without noise. This scheme is also robust for other curves, as demonstrated in the next example. We report in Figure 6.15 the initial B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse but not noisy data points, with noise level of  $\alpha = 0.0$ . For this example, the *slope-based* scheme identified 10 control points as in the last examples.

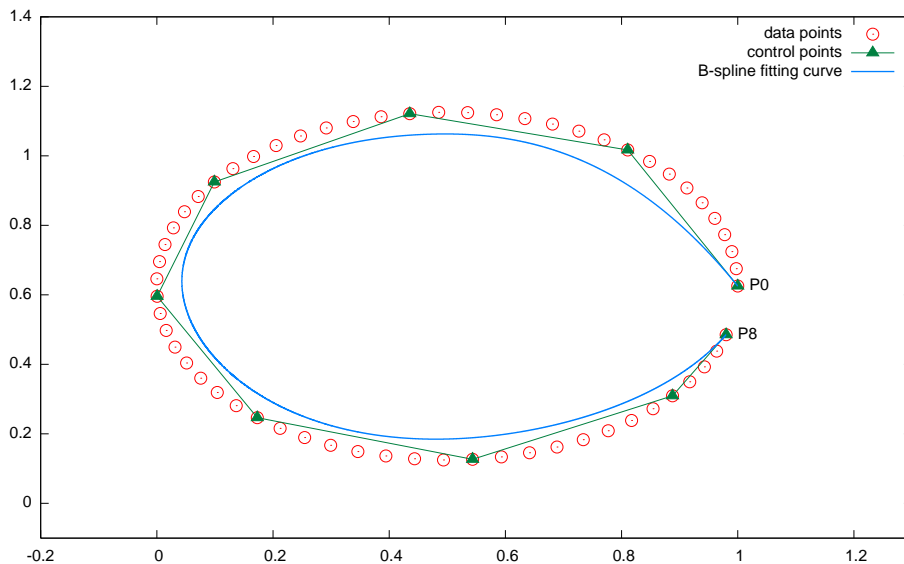


Figure 6.14: Initial solution determined through the *slope-based* control points initialization scheme, for a partial circle derived from 60 uniformly spaced and not noisy data points

In Figure 6.16, we report the comparison between the *slope-based* and the *random subset* initialization schemes. The *slope-based* initialization outperforms the *random subset* scheme, suggesting this is a promising direction. Unfortunately, this scheme doesn't have any tolerance to noise. An evident

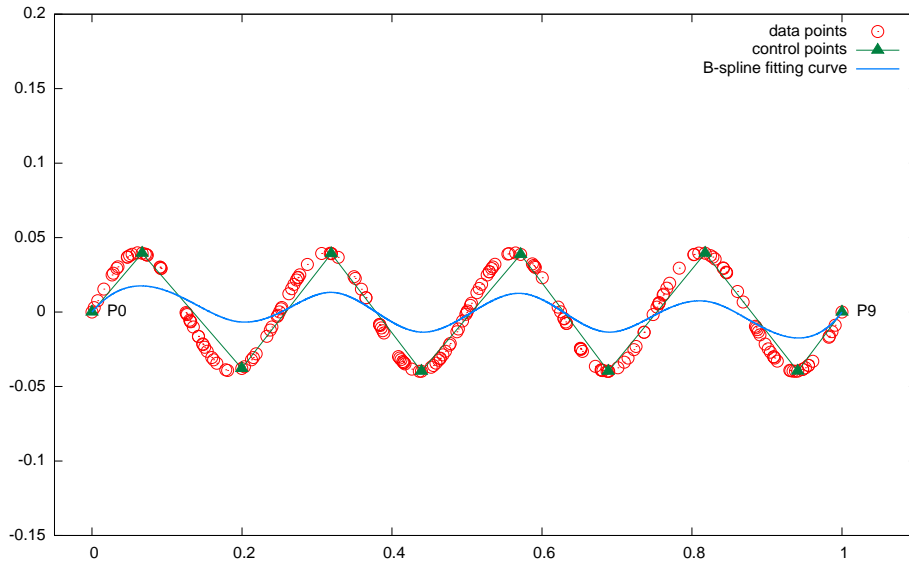


Figure 6.15: *Slope-based* initial solution of  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse but not noisy data points

example is reported in Figure 6.17, where the same curve of the last experiment is considered, with noise level of  $\alpha = 0.5$ .

Also, this approach depends on the coordinate system and so it doesn't determine the same solutions if the shape is rotated, something desirable with B-splines. In the next section, we present a more robust and accurate method, that extends the *slope-based* scheme.

### 6.5.2 Change-of-direction-based initial solution

In the previous section, we have presented the *slope-based* algorithm. The basic idea remains the same also for the *Change-of-direction-based* algorithm, we refer to as *changeDir*: We want to determine an initial B-spline fitting curve such that it coarsely approximates the shape of the target curve, picking as control points a subset of the data points. This subset is constructed by considering data points where there is a high curvature of the curve, taking into account the noise effect. Instead of comparing the angular coefficients of consecutive segments of the data points polyline, we consider here the

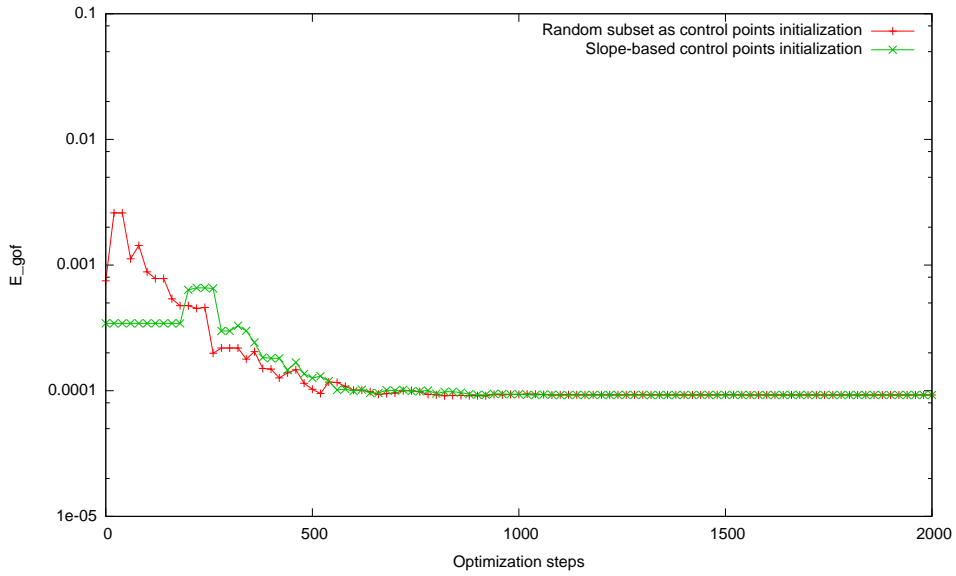


Figure 6.16: Comparison between *random subset* and *slope-based* initialization schemes of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse but not noisy data points

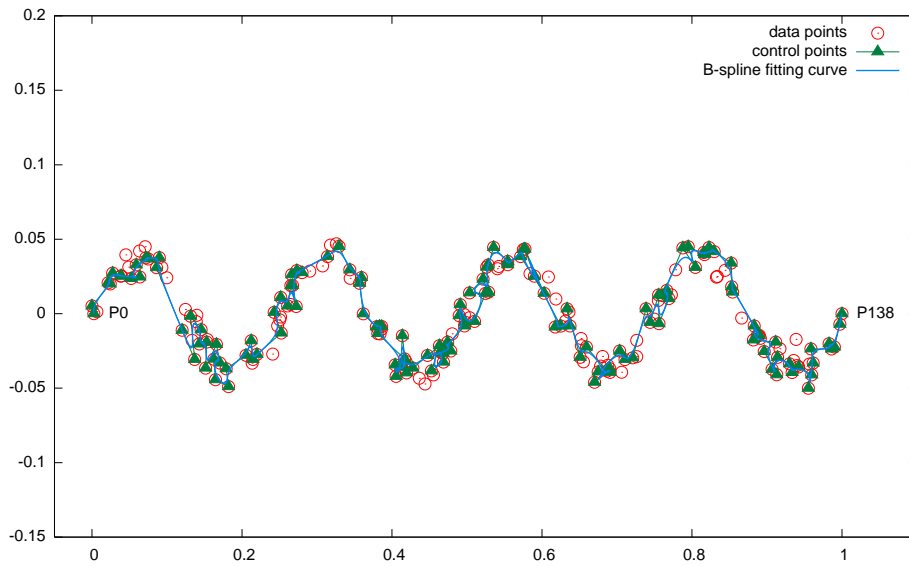


Figure 6.17: *Slope-based* initial solution of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points

---

**Algorithm 1** Determine control points for *slope-based* initial solution

---

```
1:  $n \leftarrow 0$ 
2:  $P_0 \leftarrow d_0$ 
3:  $m_1 \leftarrow 0$ 
4: for  $i \leftarrow 1$  to  $s - 1$  do
5:    $m_0 \leftarrow m_1$ 
6:    $m_1 \leftarrow (y_{d_i} - y_{d_{i-1}})/(x_{d_i} - x_{d_{i-1}})$ 
7:   if  $i = 1$  then
8:      $m_0 \leftarrow m_1$ 
9:   end if
10:  if  $(m_0 > 0 \wedge m_1 < 0) \vee (m_0 < 0 \wedge m_1 > 0) \vee (m_0 > 1 \wedge m_1 < 1) \vee (m_0 < 1 \wedge m_1 > 1) \vee (m_0 > -1 \wedge m_1 < -1) \vee (m_0 < -1 \wedge m_1 > -1)$  then
11:     $n \leftarrow n + 1$ 
12:     $P_n \leftarrow d_i$ 
13:  end if
14: end for
15:  $n \leftarrow n + 1$ 
16:  $P_n \leftarrow d_s$ 
```

---

angles they form, an explicit measure of the direction change. This approach is more robust also because it doesn't depend on any coordinate system, so operations on the data points like translation and rotation don't influence the quality of the initial solution. Similar algorithms have been proposed to measure curve similarity, as in [1]. We proceed similarly to the *slope-based* algorithm, considering any two consecutive segments identified respectively by data points  $d_{k-1}, d_k$  and  $d_k, d_{k+1}$ . We want to measure the angle they form, that determines the direction change. We can obtain it translating the first segment  $d_{k-1}d_k$  using  $d_{k-1}$  as the origin and translating the second segment  $d_k d_{k+1}$  using  $d_k$  as the origin, obtaining respectively the vectors  $a = (a_x, a_y) = (d_k - d_{k-1})$  and  $b = (b_x, b_y) = (d_{k+1} - d_k)$ . The absolute value of the angle  $\beta$  between vectors  $a, b$  is determined rearranging the dot product definition

$$\beta = \arccos \frac{a \cdot b}{|a||b|} \quad (6.8)$$

We are interested also in the angle sign: If the two consecutive segments turn to the right, the angle should be negative, while if the two consecutive segments turn to the left, the angle should be positive. The sign of the angle is equal to the sign of the cross product  $a \times b$ , which can be determined as the determinant of the matrix

$$a \times b = \begin{pmatrix} a_x & b_x \\ a_y & b_y \end{pmatrix} = a_x b_y - b_x a_y \quad (6.9)$$

At this point, we have a signed measure of the direction change at any two adjacent segments of the data points polyline. Individual direction changes may be very low. The adopted solution is to sum them up, obtaining multiple advantages: The cumulative change in direction can recognize high curvature regions of the curve, while contrasting the noise effect by balancing measurement errors in opposite directions. The noise is normally distributed by assumption to any direction and its mean is zero. We propose to place a control point every time a threshold  $\gamma$  is reached at the aggregated change in direction. We have now the problem of define properly  $\gamma$ . In presence of

sparse but not noisy data points, we can proceed similarly to the *slope-based* algorithm, considering  $\gamma = \pi/4$  which corresponds to the angle between any of the lines

$$y = 0, x = 0, y = x, y = -x \quad (6.10)$$

The complete algorithm is presented in Algorithm 2.

In Figure 6.18, we report the resulting initial solution for a set of data points derived from 60 uniformly spaced and not noisy data points. The resulting initial solution is similar to the solution determined through the *slope-based* scheme, confirming that  $\gamma = \pi/4$  is an equivalent threshold.

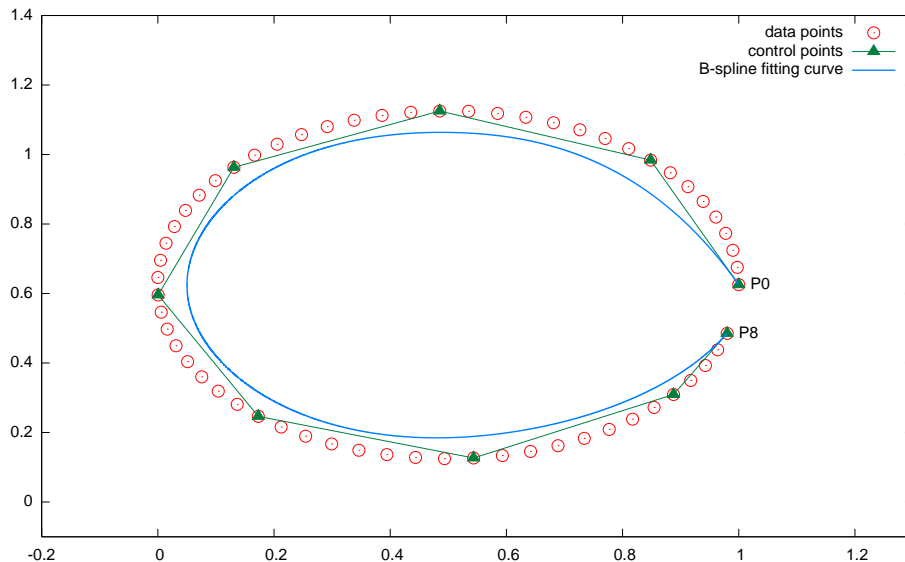


Figure 6.18: Initial solution determined through the *changeDir* control points initialization scheme, for a partial circle derived from 60 uniformly spaced and not noisy data points

This scheme is also robust to sparse data points, as demonstrated in the next example. We report in Figure 6.19 the initial B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse but not noisy data points, so with noise level of  $\alpha = 0$ . For this example, the *changeDir* scheme identified 10 control points as in the last examples. In Figure 6.20, we report

the comparison between the *slope-based*, *changeDir* and the *random subset* initialization schemes. The *changeDir* initialization outperforms the *random subset* scheme, suggesting this is a promising direction. Unfortunately, also this scheme, if applied directly to the data points, cannot filter out the noise. An evident example of this situation is reported in Figure 6.21, where the same curve of the last experiment is considered, with noise level of  $\alpha = 0.5$ . As with the *slope-based* initialization scheme presented in the previous section, nearly any data point generates a control point.

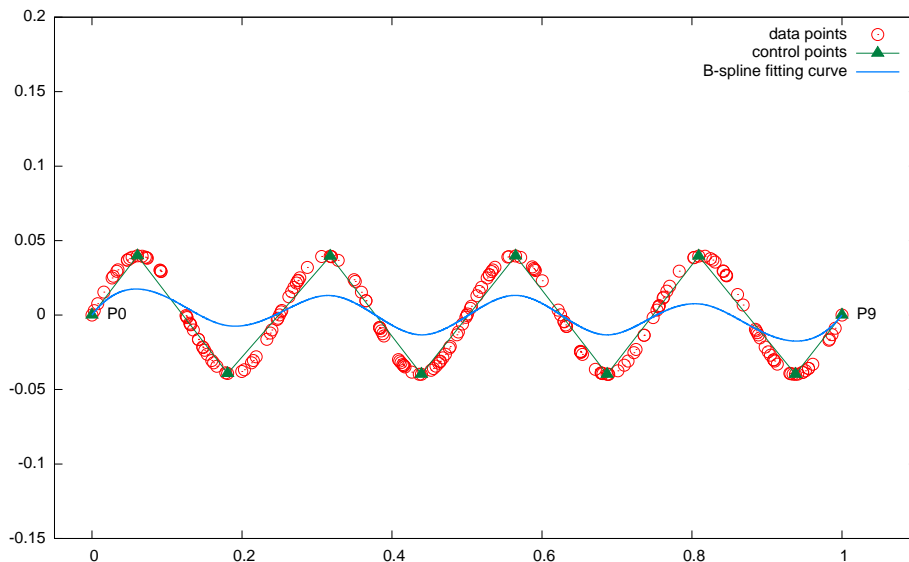


Figure 6.19: *changeDir* initial solution of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse but not noisy data points

A first attempt to filter out the noise consists of fixing the number of control points  $n + 1$  to 10 adjusting iteratively the angle  $\gamma$ , the threshold that is used by the *changeDir* scheme to determine where to place new control points. The correct value of  $\gamma$  is such that the *changeDir* algorithm determines exactly  $n + 1$  control points. If  $\gamma$  is too high, the *changeDir* scheme determines too few control points, otherwise if  $\gamma$  is too low, the *changeDir* scheme determines too many control points. We can proceed initializing  $\gamma$  to a high value, say  $2\pi$ , decreasing it until the *changeDir* scheme determines

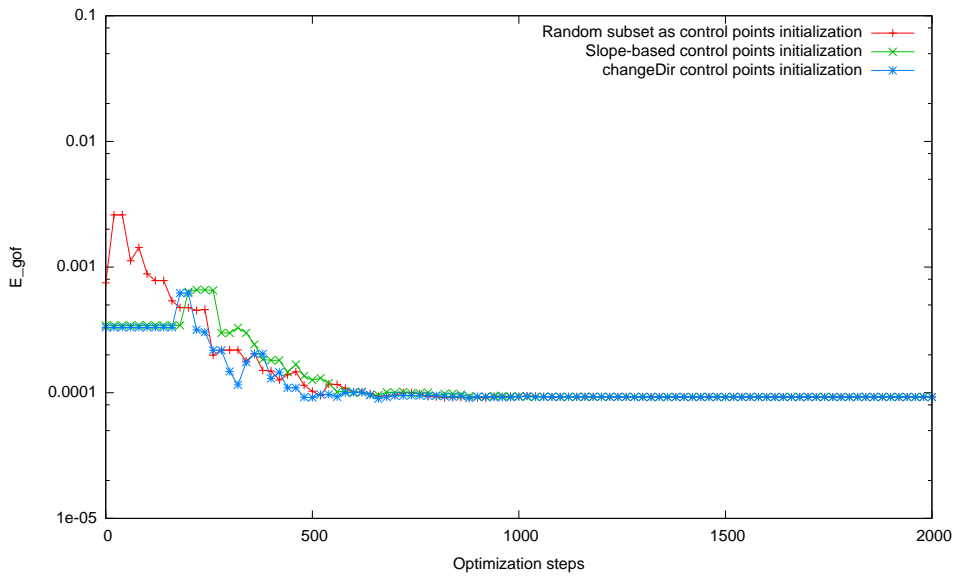


Figure 6.20: Comparison between *random subset*, *slope-based* and *changeDir* initialization schemes of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse but not noisy data points

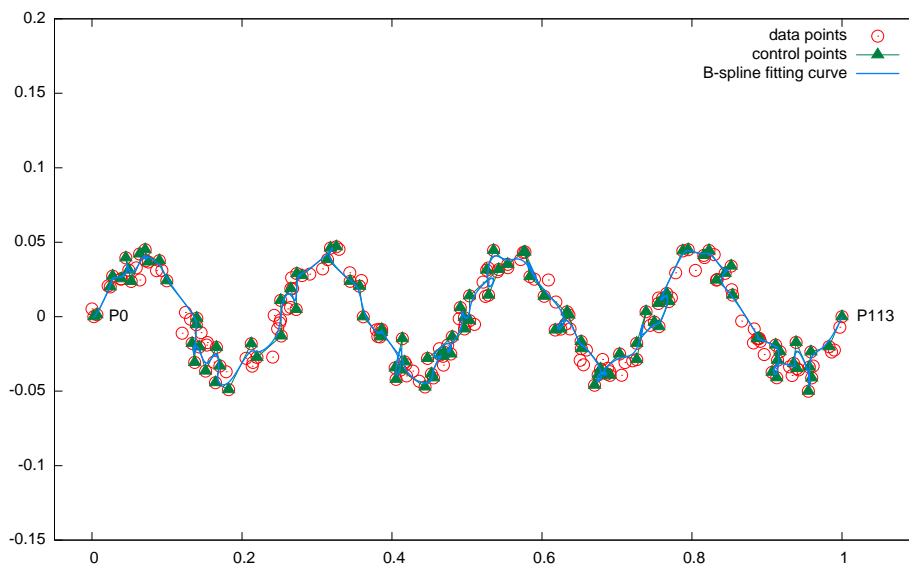


Figure 6.21: *changeDir* initial solution of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points

at least  $n + 1$  control points. This adaptive adjustment of the  $\gamma$  threshold requires many attempts, where each is an execution of the *changeDir* scheme. In Figure 6.22, we report the initial B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ , fixing  $n = 9$  and adjusting adaptively the  $\gamma$  threshold.

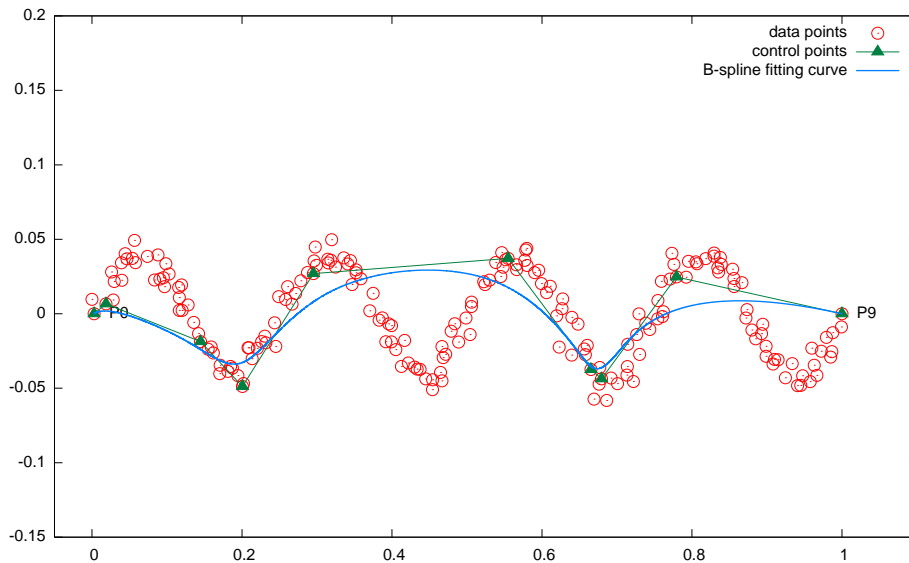


Figure 6.22: Initial solution determined through the *changeDir* control points initialization scheme of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, fixing  $n = 9$  and adjusting accordingly the threshold  $\gamma$

This approach unfortunately doesn't work, because in presence of noise and dense data points, as in the reported example, we have that adjacent data points lead to very high direction changes. When this situation occurs, the threshold  $\gamma$  is reached and a control point placed. We haven't any control on those situations, that are encountered randomly from the first to the last data point. The adaptive choice of the  $\gamma$  threshold reduces the *changeDir* initialization scheme to the *random subset* scheme, where we pick a subset of random data points. This experiment suggests that the noise should be filtered out independently of direction changes. In the next sec-

tion, we propose to introduce preprocessing techniques to filter out the noise, in particular through *sampling* and *smoothing*.

### 6.5.3 Data preprocessing by sampling and smoothing

We propose here to reduce the perverse effect of noise through preprocessing techniques, in particular combining *sampling* and *smoothing*. The *sampling* procedure consists of reducing the series of data points  $d_0, \dots, d_s$  to the series  $d_0, d_\delta, d_{2\delta}, \dots, d_{k\delta}, d_s$ , picking a data point every  $\delta$  data points, plus the first and last data points. Note that the noise effect on the direction change of two adjacent data points  $d_i, d_{i+1}$  is inversely proportional to their distance  $|d_i - d_{i+1}|$ . Figure 6.23 shows an example. When the distance between the two data points is greater, the noise effect is lower.

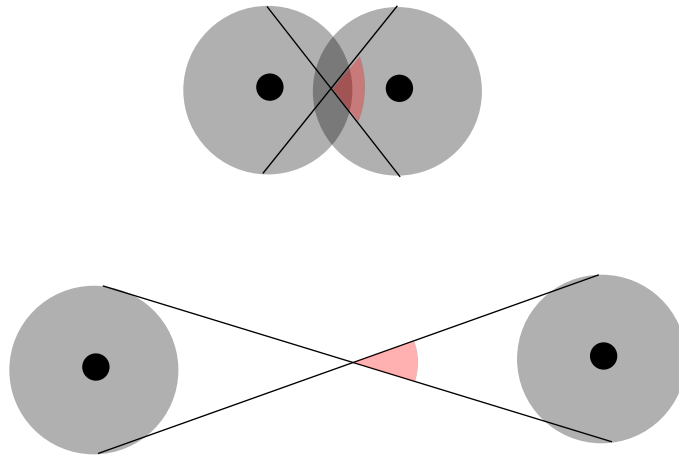


Figure 6.23: Example of noise effect on direction change of two adjacent data points: the change of direction caused by noise decreases when points are more distant

The average distance between the sampled data points  $d_0, d_\delta, d_{2\delta}, \dots, d_{k\delta}, d_s$  is greater than the average distance between the series of data points  $d_0, \dots, d_s$ , so this property allows us to reduce the effect of noise on direction changes. The *sampling* step  $\delta$  must be chosen carefully: If it is too high, the picked

data points don't capture the shape of the curve, if it is too low, we keep the noise effect high.

This algorithm, applied to the data points reported in Figure 6.24, leads to the data points subset reported in Figure 6.25. Reducing the data points density, we have also reduced the noise effect on direction changes.

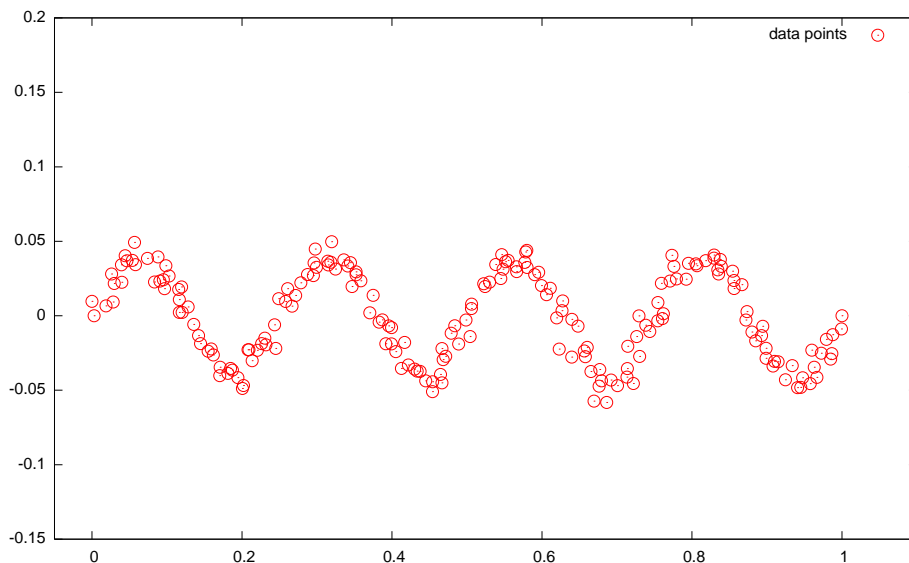


Figure 6.24:  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points

In presence of many consecutive data points grouped in a small region, *sampling* is less effective in contrasting the noise effect on direction changes. To address this problem, we propose here to introduce *smoothing*. The *smoothing* procedure applied after *sampling* consists in averaging the data points  $d_0, d_\delta, d_{2\delta}, \dots, d_{k\delta}, d_s$  using the complete data points series  $d_0, \dots, d_s$ , considering their adjacent data points. For example, the data point  $d_\delta$  is substituted with the averaged point  $d_{\delta-\varphi}, \dots, d_\delta, \dots, d_{\delta+\varphi}$ , where  $\varphi$  identifies the size of the average. Let's recall that the noise is normally distributed, so averaging consecutive data points allows us to compensate errors in opposite directions. This *smoothing* property allows us to address the noise effect also when *sampling* performs poorly. Their combination results in a more robust

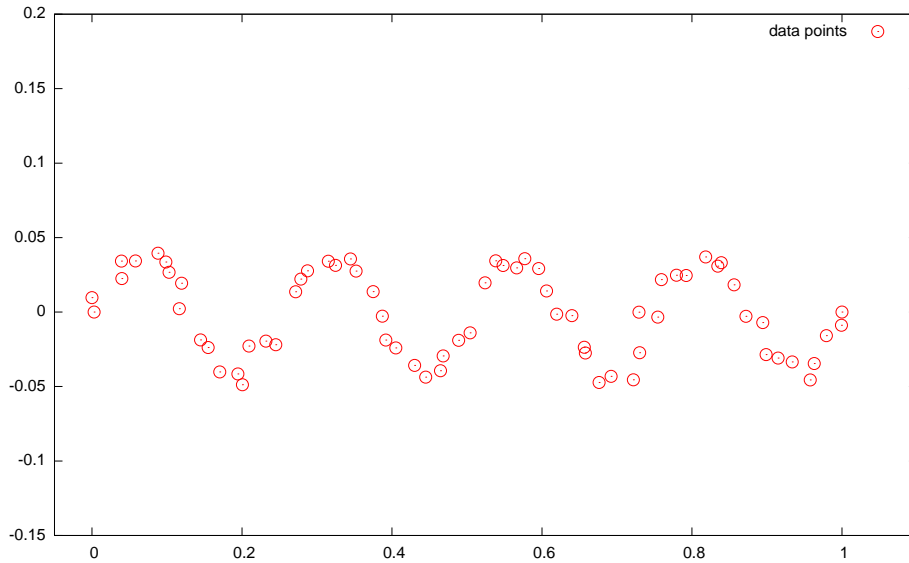


Figure 6.25:  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with *sampling*

solution.

This algorithm, applied to the data points of the example in Figure 6.25 with  $\delta = 3, \varphi = 5$ , leads to the smoothed data points reported in Figure 6.26. Figure 6.27 shows the corresponding initial B-spline curve. In this test, the *changeDir* scheme identified still 10 control points, so  $n = 9$ . As desired, the combined *sampling* and *smoothing* procedures applied to the *changeDir* initialization scheme contrasts sufficiently the noise effect.

Figure 6.28 shows the comparison between the *changeDir* and the *random subset* initialization schemes, both with *sampling* and *smoothing* configured with  $\delta = 3, \varphi = 5$ . The *changeDir* initialization outperforms the *random subset* scheme, also with noisy data points. We report in Figure 6.29 the resulting B-spline fitting curve after 2000 optimization steps, adopting the *changeDir* initialization with *sampling* and *smoothing*.

The choice of  $\delta$  and  $\varphi$  is empirical. Further investigation considering partial knowledge about the process of noise generation, or estimating the noise model through cross-validation are considered as future research directions.

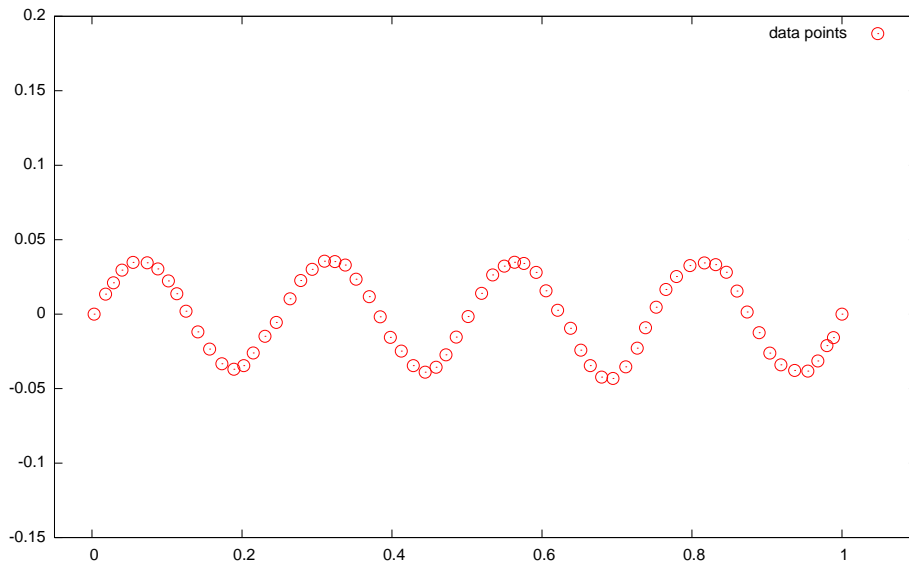


Figure 6.26:  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with *sampling* and *smoothing*

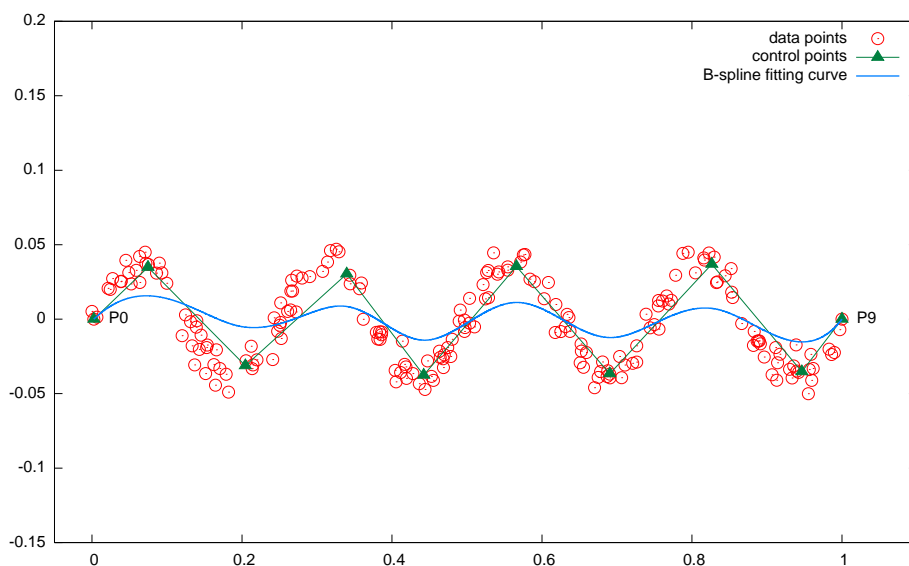


Figure 6.27: *changeDir* initial solution of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with *sampling* and *smoothing*.

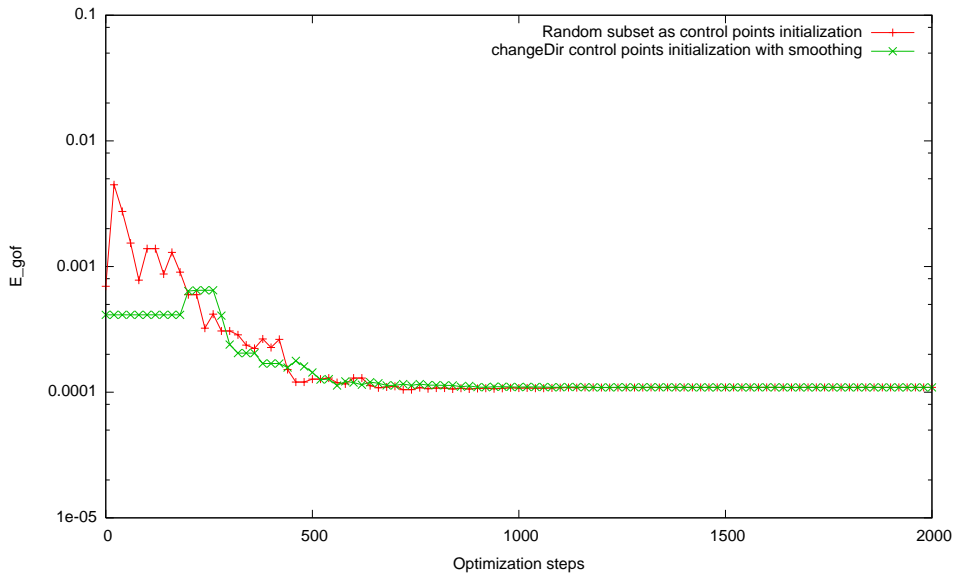


Figure 6.28: Comparison between *random subset* and *changeDir* initialization schemes of  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse and noisy data points

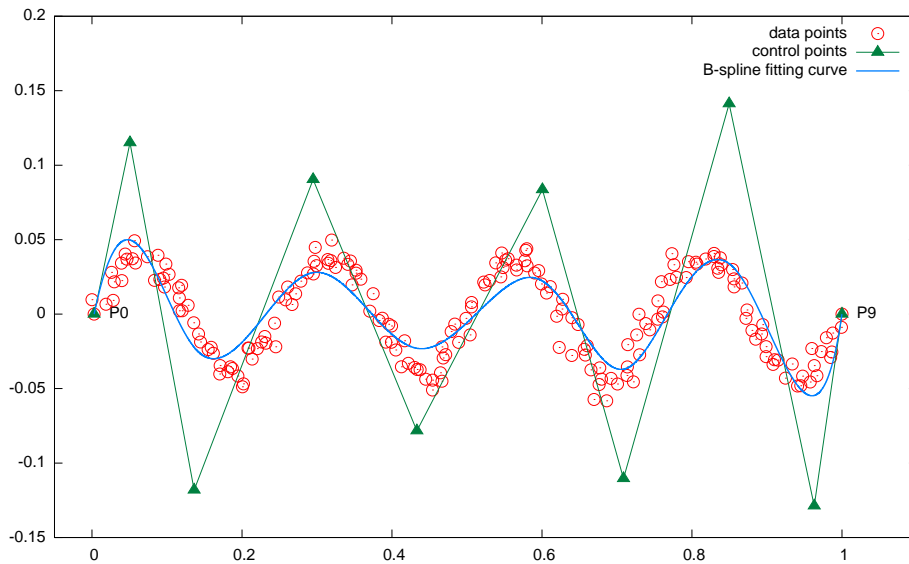


Figure 6.29: B-spline fitting curve of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, adopting the *changeDir* initial solution with *sampling* and *smoothing*, after 2000 optimization steps

Those techniques influence only the choice of the number of control points, and their initial positioning. During the optimization process, we consider all the data points, without *sampling* and *smoothing*.

We present now another experiment. Figure 6.30, Figure 6.31, Figure 6.32 show the histogram of  $E_{gof}$  frequencies for different initialization schemes after fitting the  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse and noisy data points, considering 2000 optimization steps. For each configuration, we have 200 runs of the optimization process. Surprisingly, the initial positioning of control points doesn't influence significantly the quality of the final solution, affecting only the convergence speed to a local minimum. This test demonstrates the robustness of the RASH optimization scheme.

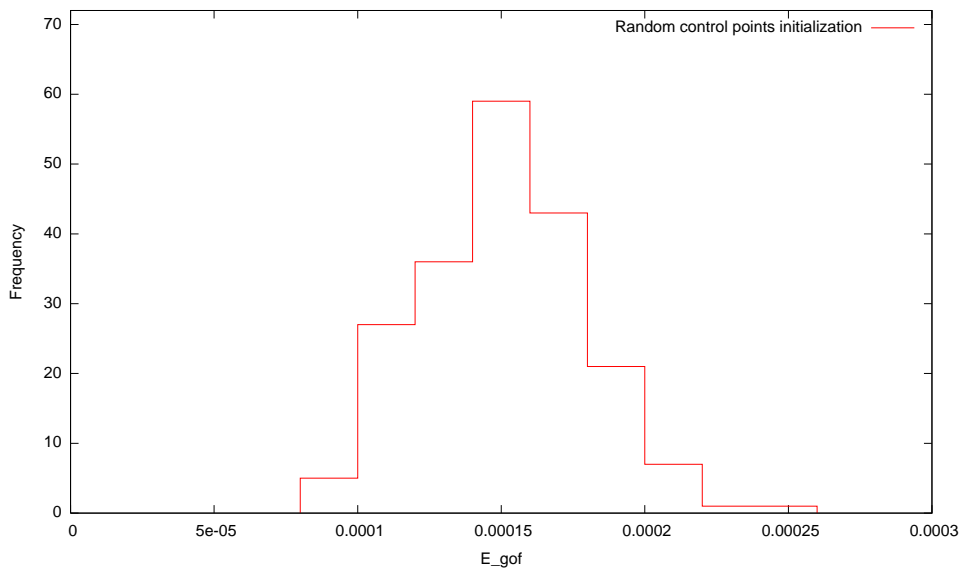


Figure 6.30: Histogram of  $E_{gof}$  frequencies for *random* initialization scheme obtained after fitting the  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse data points, considering 2000 optimization steps. We have 200 runs of the optimization process.

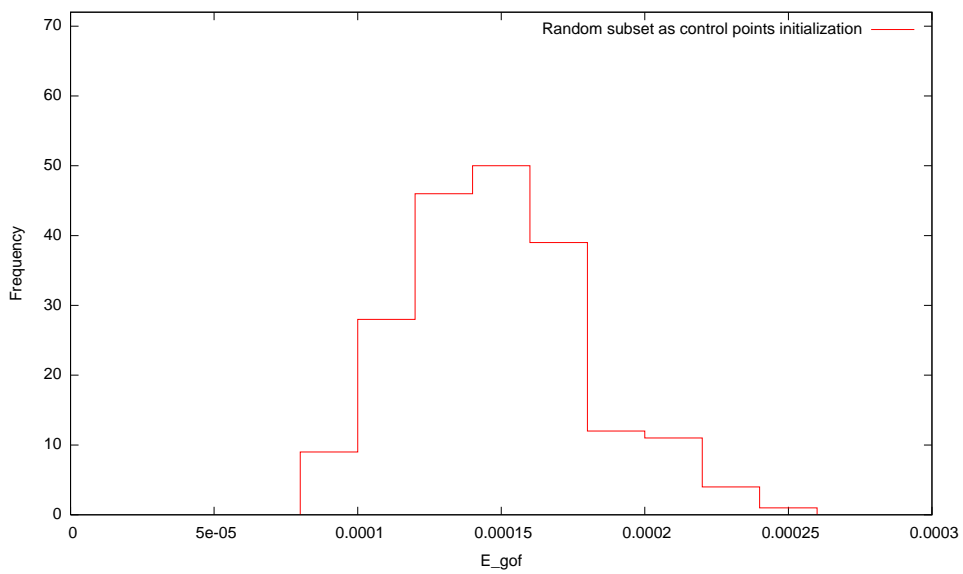


Figure 6.31: Histogram of  $E_{gof}$  frequencies for *random subset* initialization scheme obtained after fitting the  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse data points, considering 2000 optimization steps. We have 200 runs of the optimization process.

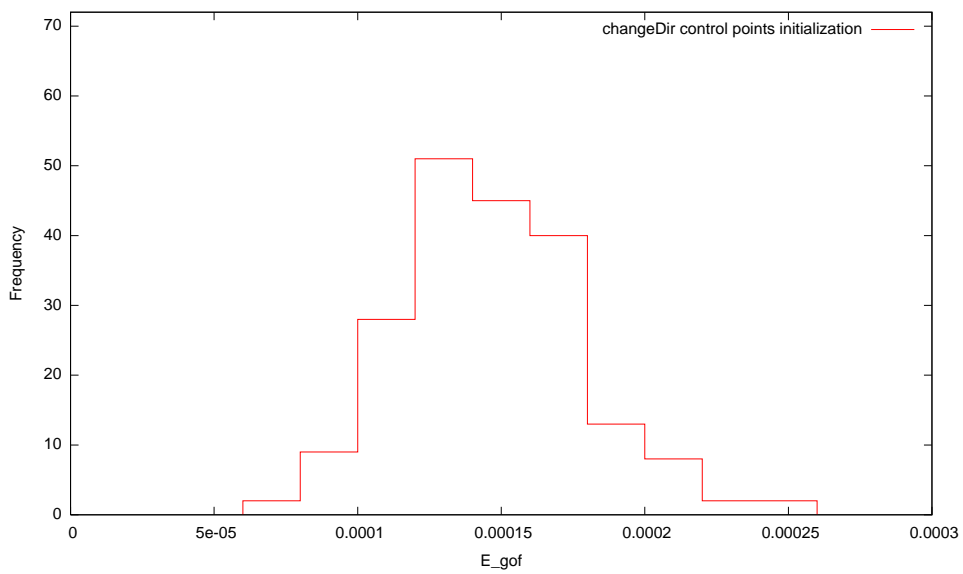


Figure 6.32: Histogram of  $E_{gof}$  frequencies for *changeDir* initialization scheme obtained after fitting the  $\sin(x)$  curve in interval  $[0, 8\pi]$ , derived from 200 sparse data points, considering 2000 optimization steps. We have 200 runs of the optimization process.

---

**Algorithm 2** Determine control points for *changeDir* initial solution

---

```
1:  $n \leftarrow 0$ 
2:  $P_n \leftarrow d_0$ 
3:  $t \leftarrow 0$ 
4: for  $i \leftarrow 1$  to  $s - 1$  do
5:    $a \leftarrow d_i - d_{i-1}$ 
6:    $b \leftarrow d_{i+1} - d_i$ 
7:    $\beta \leftarrow a \cdot b / |a||b|$ 
8:   if  $a \times b < 0$  then
9:      $\beta \leftarrow -\beta$ 
10:  end if
11:   $t \leftarrow t + \beta$ 
12:  if  $t > \pi/4$  then
13:     $n \leftarrow n + 1$ 
14:     $P_n \leftarrow d_i$ 
15:     $t \leftarrow 0$ 
16:  end if
17: end for
18:  $n \leftarrow n + 1$ 
19:  $P_n \leftarrow d_s$ 
```

---

# Chapter 7

## The parametrization problem

Once we have determined the number and the initial positioning of the  $n - 1$  internal control points that coarsely approximates the shape of the target curve, we can proceed by associating each data point  $d_k$  to a point on the  $C$  curve. This procedure is referred to as parametrization and consists of associating the data points  $d_0, \dots, d_s$  to points  $C(t_0), \dots, C(t_s)$  on the  $C$  curve, where  $t_0, \dots, t_s$  are the parameters. Parametrization is not part of the B-spline curve definition, but it is used in the objective function to minimize in order to approximate efficiently  $E_{gof}$  with  $E_{param}$ . Once the parameters are determined, we have all information for the objective function to minimize. The classical parametrization schemes *uniform*, *chord-length* and *centripetal* have been presented in Chapter 2. Here, we propose new parametrization schemes and discuss also their possible combination in a hybrid approach.

### 7.1 The global minimum distance parametrization

Let's recall that we want to minimize the distance between each data point  $d_k$  and its respective approximated value on the B-spline fitting curve,  $C(t_k)$ . We propose to associate to each data point  $d_k$  a point on the B-spline curve that has minimum distance from the given point. The parameter is then

derived immediately. The definition of the parametrization problem is relaxed in the sense that if data point  $d_k$  follows data point  $d_{k-1}$ , we don't require that  $t_k \geq t_{k-1}$ . By definition, parameters determined with the *global minimum distance* parametrization minimize the sum of squared distances between data points and their corresponding points on the B-spline curve  $C$ . The objective function value  $E_{param}$  adopting the *global minimum distance* parametrization scheme is a close approximation of  $E_{gof}$  if a sufficiently large number of parameter values is tested to determine the approximated minimum distance. The *global minimum distance* algorithm is reported in Algorithm 3. This parametrization technique requires to sample the B-spline curve  $C$ , so when the  $C$  curve changes, we must update the parameters. Sampling at each optimization step the  $C$  curve is computationally expensive, so we can approximate the parameters and update them only when the objective function value decreases by a certain threshold. Note that if the objective function value changes significantly, also the B-spline fitting curve changed significantly. This scheme is adaptive in the sense that parameters are not fixed but are updated during the optimization process.

Figure 7.1 shows the comparison between the *centripetal* and *global minimum distance* parametrization schemes for the B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ . The initial solution has been determined with the *changeDir* scheme with *sampling* and *smoothing*. As you can see, the *global minimum distance* parametrization performs better than the *centripetal* parametrization scheme.

## 7.2 The local minimum distance parametrization

The *global minimum distance* parametrization doesn't consider the monotonicity property of the parametrization function, which associates to any data point  $d_k$  its respective parameter  $t_k$ . The monotonic property states

---

**Algorithm 3** Determine *global minimum distance* parameters

---

```
1: for  $i \leftarrow 0$  to  $s$  do
2:    $d[i] \leftarrow \infty$ 
3:    $t_i \leftarrow 0$ 
4: end for
5:  $u \leftarrow 0$ 
6: while  $u \leq 1$  do
7:    $c \leftarrow C(u)$ 
8:   for  $i \leftarrow 0$  to  $s$  do
9:      $d \leftarrow \sqrt{(x_{d_i} - x_c)^2 + (y_{d_i} - y_c)^2}$ 
10:    if  $d \leq d[i]$  then
11:       $d[i] \leftarrow d$ 
12:       $t_i \leftarrow u$ 
13:    end if
14:  end for
15:   $u \leftarrow u + \delta$ 
16: end while
17:  $t_0 \leftarrow 0$ 
18:  $t_s \leftarrow 1$ 
```

---

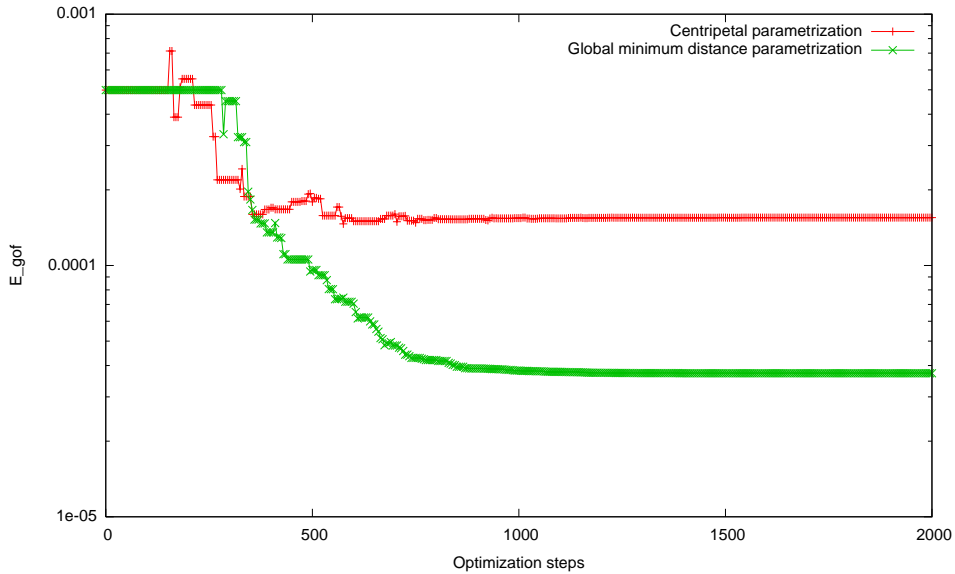


Figure 7.1: Comparison between *centripetal* and *global minimum distance* parametrization schemes, of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points

that for any  $k$  in  $0, \dots, s - 1$

$$t_k \leq t_{k+1} \tag{7.1}$$

We observe that if this property holds for a particular instance of the *global minimum distance* parametrization, the resulting parameters are optimal. This anyway isn't true in general. We propose here to pick local optimum parameters, such that they minimize their distance between the respective data points locally and respecting the monotonicity property. In details, we proceed as follows. In order to determine the parameter for data point  $d_k$ , we consider as feasible parameter any parameter value between the parameter associated to the previous data point,  $t_{k-1}$ , and the last feasible parameter associated to the last data point, 1:  $[t_{k-1}, 1]$ . In this interval, we take the parameter  $t_k$  which leads to the first local minimum, where we try to minimize the distance between  $d_k$  and  $C(t_k)$ . This model leads to the algorithm presented in Algorithm 4.

---

**Algorithm 4** Determine *local minimum distance* parameters

---

```
1:  $mind \leftarrow \infty$ 
2:  $minu \leftarrow 0$ 
3: for  $i \leftarrow 0$  to  $s$  do
4:    $t_i \leftarrow 1$ 
5: end for
6:  $i \leftarrow 0$ 
7:  $u \leftarrow 0$ 
8: while  $u \leq 1$  do
9:    $c \leftarrow C(u)$ 
10:   $d \leftarrow \sqrt{(x_{d_i} - x_c)^2 + (y_{d_i} - y_c)^2}$ 
11:  if  $d < mind$  then
12:     $mind \leftarrow d$ 
13:     $minu \leftarrow u$ 
14:  else
15:     $t_i \leftarrow minu$ 
16:     $i \leftarrow i + 1$ 
17:     $mind \leftarrow \infty$ 
18:  end if
19:   $u \leftarrow u + \delta$ 
20: end while
21:  $t_0 \leftarrow 0$ 
22:  $t_s \leftarrow 1$ 
```

---

The proposed algorithm samples the B-spline fitting curve  $C$  from 0 to 1, picking parameters for the data points in ascending order  $d_0, \dots, d_s$ . We can define also its opposite, which samples  $C$  from 1 to 0 picking parameters for data points in descending order  $d_s, \dots, d_0$ . Note that if the parameters found through the two versions of this algorithm are the same, those parameters are also optimal. Anyway, this is not true in general.

In Figure 7.2, we report a comparison between *centripetal*, *global minimum distance* and *local minimum distance* parametrization schemes for the B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ . The initial solution has been determined through the *changeDir* scheme. The *local minimum distance* parametrization performs worse than the *global minimum distance*, but wins against the *centripetal* scheme. This suggests that the first encountered local minimum for the distance between the B-spline curve and each data point is good but it is not also the global minimum.

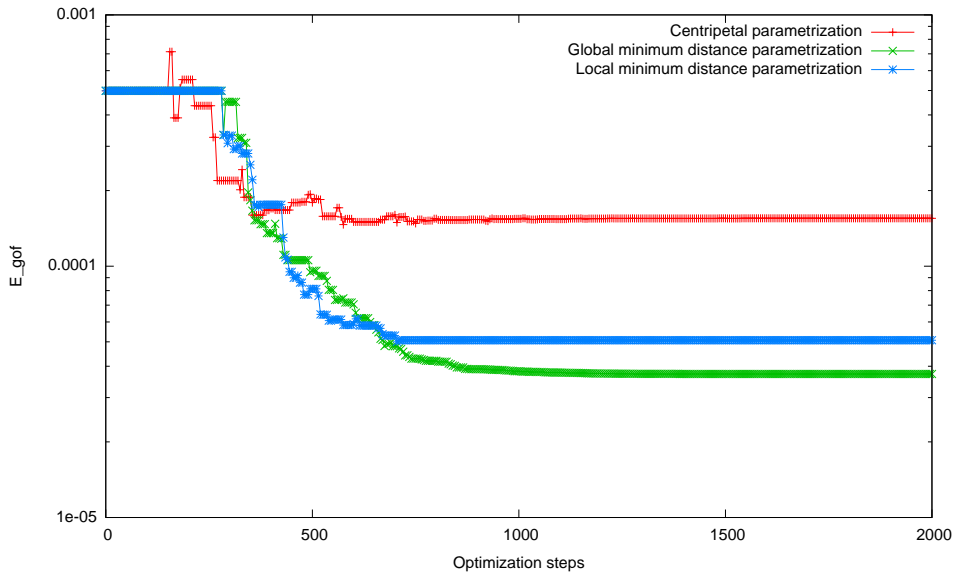


Figure 7.2: Comparison between *centripetal*, *global minimum distance* and *local minimum distance* parametrization schemes, of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points

We present now an experiment on the robustness of the proposed parametrization schemes. In Figure 7.3, we report a comparison of  $E_{gof}$  for different parametrization schemes in the first 8000 optimization steps for the B-spline fitting curve of the  $\sin(x)$  curve, in interval  $[0, 8\pi]$ , derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ . The initial solution is chosen with the *random* scheme. This preliminary test indicates that the *global minimum distance* and *local minimum distance* parametrization schemes are very sensitive to the initial solution: These parametrization schemes are unable to untangle the initial B-spline curve as the *centripetal* scheme, based on the arc-length.

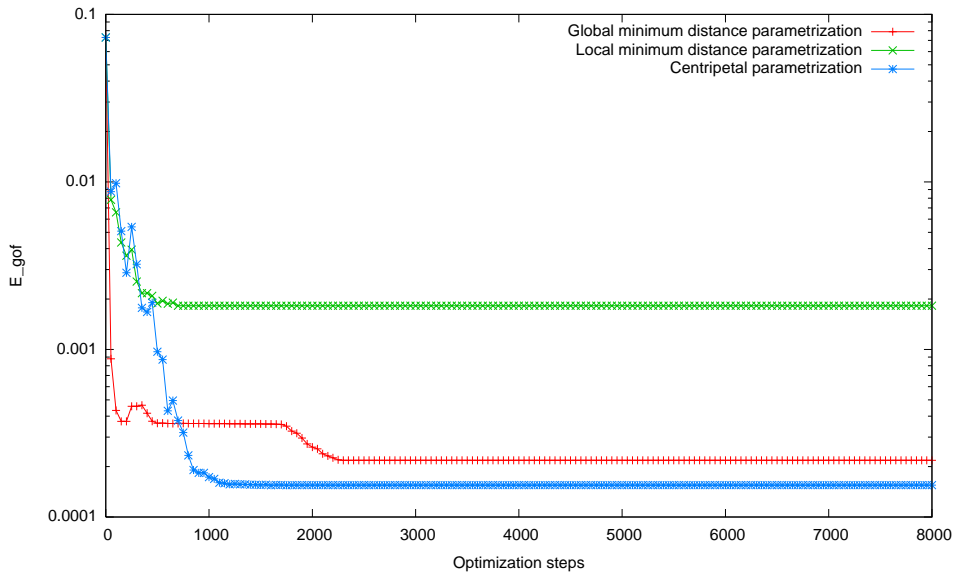


Figure 7.3: Comparison of  $E_{gof}$  in the first 8000 optimization steps for the B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ . The initial solution is chosen with the *random* scheme

### 7.3 The adaptive hybrid parametrization scheme

The *global minimum distance* parametrization performs better than the *centripetal* parametrization scheme, but it is computationally more expensive because the parameters must be updated also during the optimization process. We propose here to combine the *centripetal* and the *global minimum distance* parametrization schemes, as follows. Initially, the *centripetal* scheme is adopted. After a certain number of steps, or when the improvements in the objective function evaluations are under a certain threshold, we switch to the *global minimum distance* parametrization scheme. Figure 7.4 shows how this technique performs for the B-spline fitting curve of the  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points, with noise level of  $\alpha = 0.5$ .

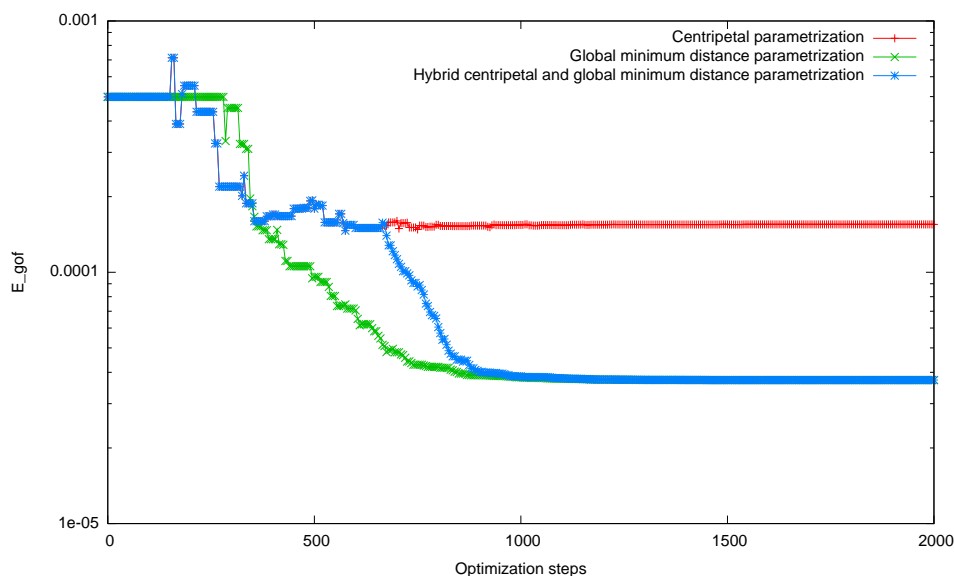


Figure 7.4: Comparison between *centripetal*, *global minimum distance* and their hybrid combination, of  $\sin(x)$  curve in interval  $[0, 8\pi]$  derived from 200 sparse and noisy data points

# Chapter 8

## Conclusions

In this work, we face the general problem of reconstructing a curve from a series of data points in the presence of noise. We require the fitting curve to be continuous until the second derivative and to pass exactly through the first and the last data points. We don't make any other particular assumption on the phenomena generating the experimental data points, and adopt cubic B-splines as the curve model. Cubic B-splines are widely adopted in computer vision [9] and signal processing [17], thanks to their capacity to fit complex parametric curves. We adopted a least squares fitting formulation, minimizing the sum of the squares of the offsets ("the residuals") of the data points to the B-spline fitting curve. Cubic B-splines are fully determined by their knots vector, which we fixed uniformly, and their control points distribution, which we have considered as variables of the optimization process. The main issues in fitting B-splines are the parametrization, the choice of the number of control points and their initial distribution. The main contributions of this work can be summarized as follows.

Instead of using standard numerical least squares methods, we adopted RASH [6], an adaptive random search algorithm which does not require derivatives. The results demonstrate that the RASH optimizer is surprisingly robust, converging to a good local minimum independently of the initial control points assignment. In the framework of Reactive Search Optimiza-

tion, which follows the paradigm of "learning while solving", we contributed different heuristics that can determine adaptively the number and the initial control points assignment, a problem that has not yet been covered sufficiently in the existing literature. Our tests show that the quality of the initial control points assignment speeds up the optimization process but it doesn't improve the quality of the final result. We proposed also some new adaptive parametrization techniques, showing that a hybrid approach can outperform the basic *centripetal* scheme while preserving a low computational cost. The tests suggest that, while a good initial B-spline fitting curve doesn't improve significantly the quality of the final solution, a better parametrization can boost it to a lower local minimum.

## 8.1 Future research directions

In this section, we discuss the open problems of our approach, giving some hints for further research. The knots vector can be adjusted adaptively during the optimization process, as we proposed for the parameters. The introduction of new variables to optimize can improve the quality of the final result, but it will probably slow down the convergence toward the local minimum. The least squares fitting formulation presented in Chapter 3 is known to be poorly tolerant to outliers, because a single big difference introduced by an outlying point has a great influence on the sum of squares of the offsets. More robust techniques may be adopted, improving the robustness of the fit. The optimizer speed can be boosted by adaptively restricting the regions of possible values for control points near their initial value. Now the intervals are fixed empirically. This idea, even if appealing, is risky because an erroneous initial placement of control points may lead to poor fits. Data preprocessing through *sampling* and *smoothing* introduced for the adaptive initialization schemes requires to tune empirically the parameters  $\delta$  and  $\varphi$ . Further investigations are possible, by using the tools of statistics and machine learning, for example one could consider the available knowledge about the process of

noise generation for statistical hypothesis testing, one could insert *a priori* knowledge about the possible trajectories through Bayesian techniques, or one could aim at estimating the noise model through cross-validation techniques.

# Appendix A

## Some complex examples

In Figure A.1, Figure A.2, Figure A.3, Figure A.4, Figure A.5, Figure A.6, Figure A.7 we report the fitting B-splines of some complex shapes.

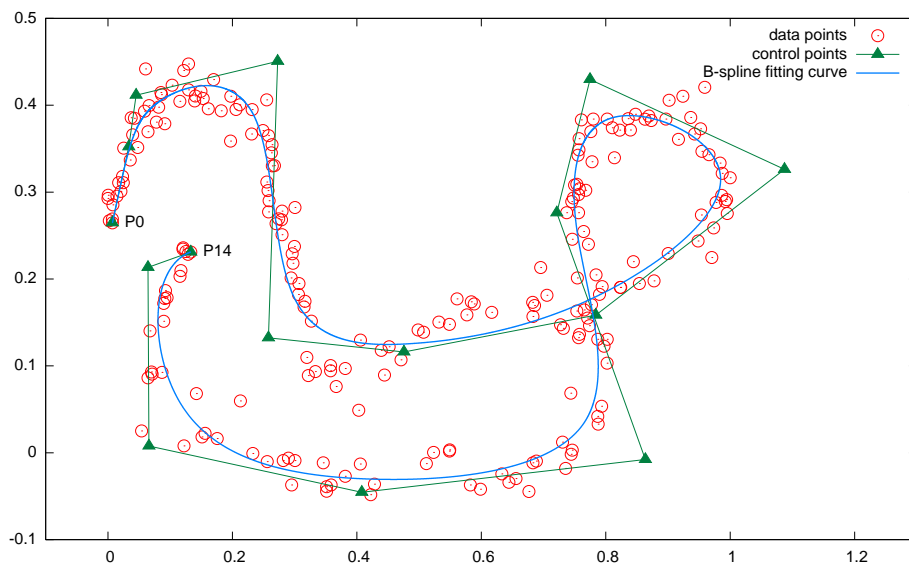


Figure A.1: B-spline fitting curve of self intersecting curve

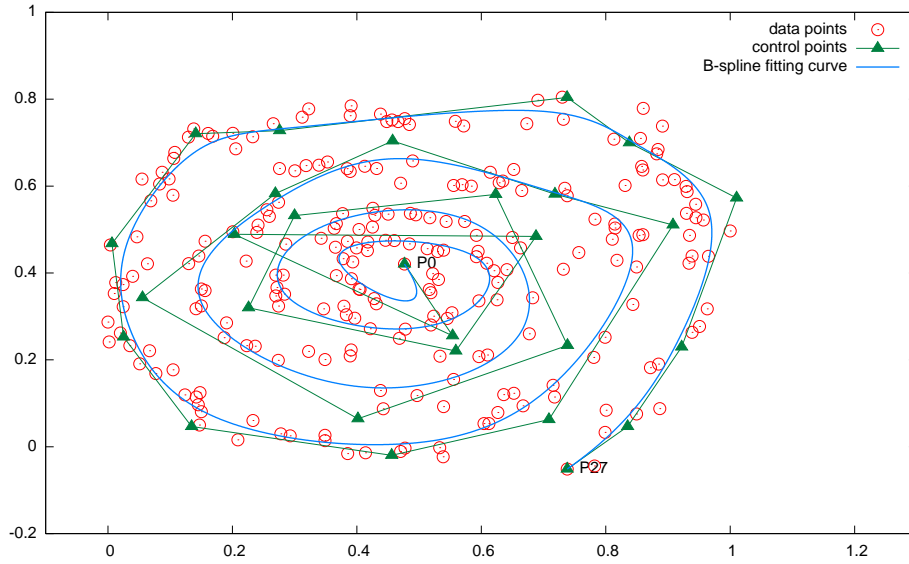


Figure A.2: B-spline fitting curve of spiral shape

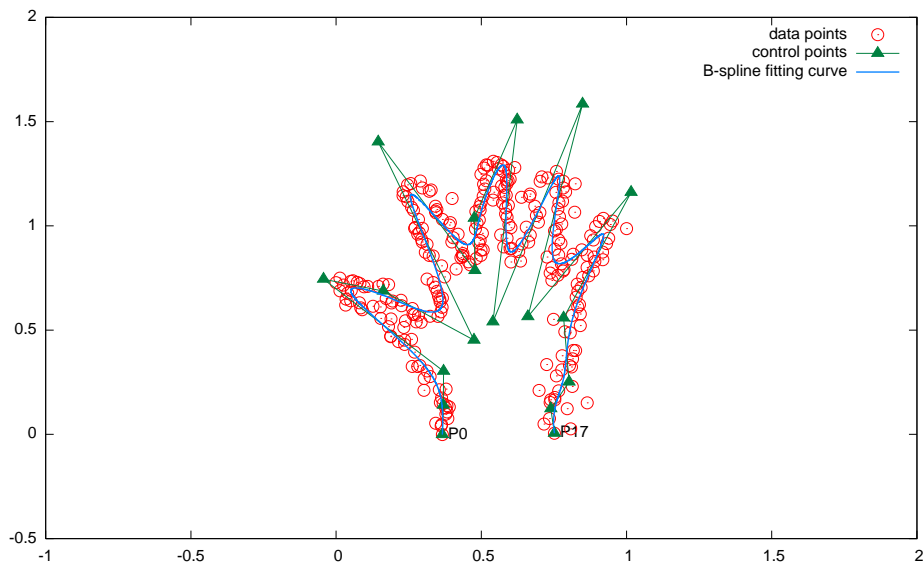


Figure A.3: B-spline fitting curve of hand shape

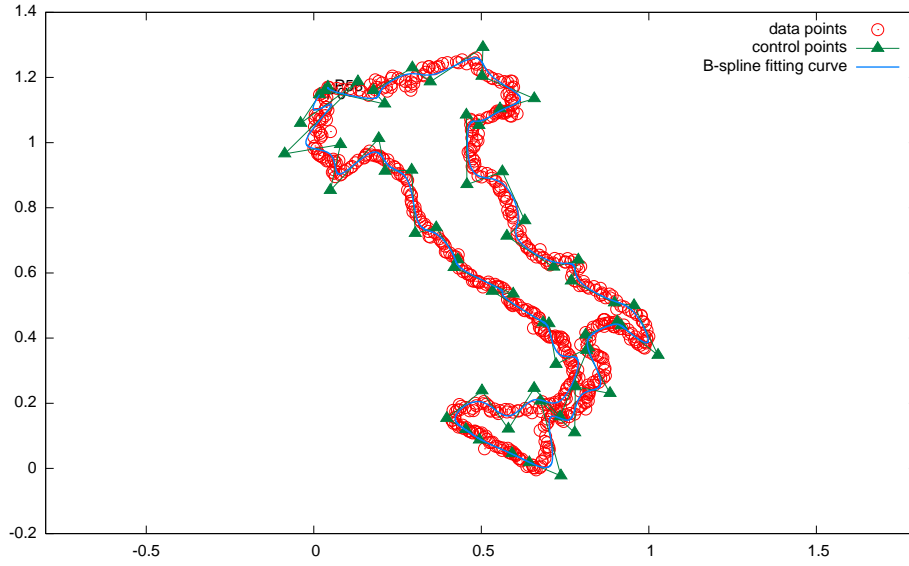


Figure A.4: B-spline fitting curve of Italy shape

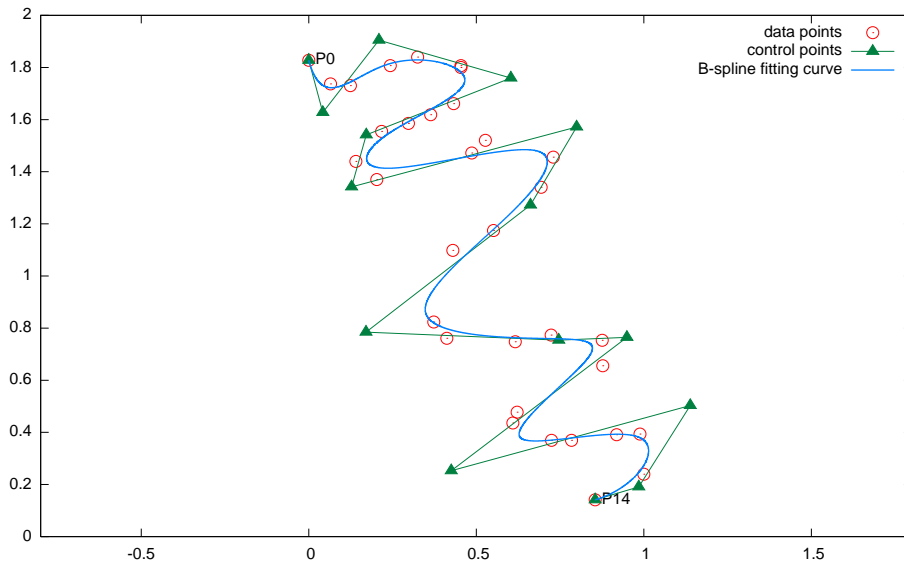


Figure A.5: B-spline fitting curve of sinusoid curve

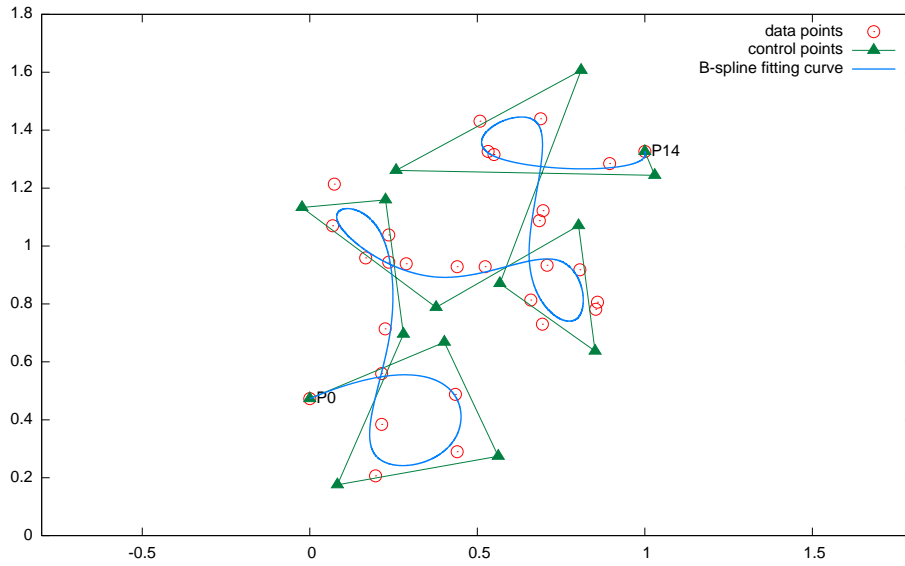


Figure A.6: B-spline fitting curve of self intersecting curve

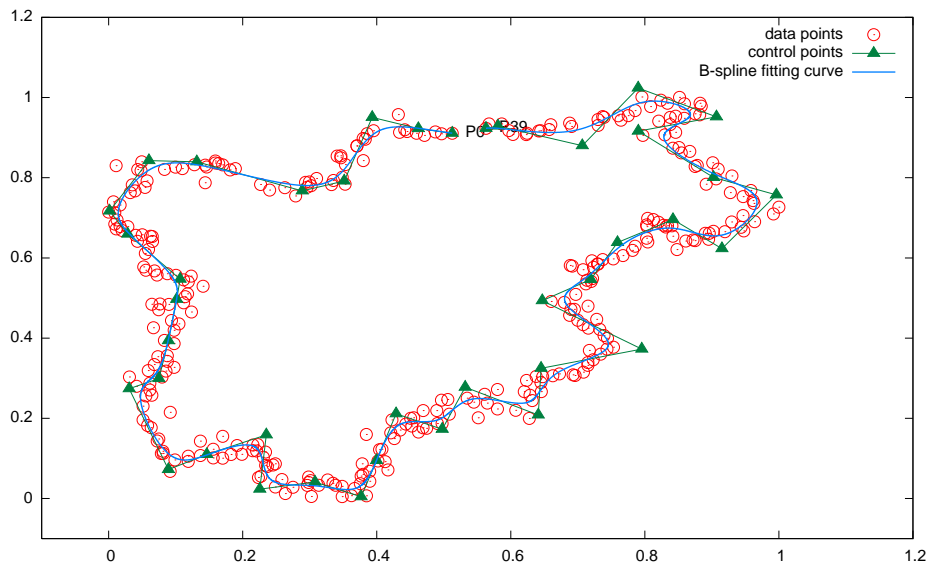


Figure A.7: B-spline fitting curve of Trentino shape

# List of Figures

2.1	Physical spline . . . . .	6
2.2	B-spline curve with explicit knots distribution . . . . .	8
2.3	Uniform parameters of a B-spline fitting curve of $\sin(x)$ curve in interval $[0, 4\pi]$ , derived from 40 uniformly spaced data points	12
3.1	The Affine Shaker algorithm . . . . .	21
3.2	Affine Shaker geometry: two search trajectories leading to two different local minima. The evolution of the search regions is also illustrated . . . . .	23
4.1	The problem of fitting with a B-spline a set of data points . . .	26
4.2	B-spline approximated curve determined through the opti- mization process . . . . .	29
5.1	Data points derived sampling the sinusoid $\sin(x)$ in interval $[0, 6\pi]$ with $\alpha = 0$ . . . . .	32
5.2	Data points derived sampling the sinusoid $\sin(x)$ in interval $[0, 6\pi]$ with $\alpha = 0.5$ . . . . .	33
6.1	B-spline fitting curve of the $\sin(x)$ curve in interval $[0, 4\pi]$ , derived from 100 sparse and noisy data points, after 2000 op- timization steps . . . . .	37
6.2	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 0 optimization steps . . . . .	38

6.3	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 200 optimization steps . . . . .	38
6.4	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 400 optimization steps . . . . .	39
6.5	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 600 optimization steps . . . . .	39
6.6	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 800 optimization steps . . . . .	40
6.7	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 1000 optimization steps . . . . .	40
6.8	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ after 2000 optimization steps . . . . .	41
6.9	Comparison of $E_{gof}$ and $E_{param}$ for B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ , 2000 optimization steps, <i>random</i> control points initialization . . . . .	41
6.10	Initial solution determined picking a <i>random subset</i> of data points as initial control points . . . . .	43
6.11	Comparison between <i>random</i> and <i>random subset</i> initialization schemes of $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse and noisy data points . . . . .	43
6.12	Initial solution determined through uniformly spaced control points along the segment connecting the two extreme data points . . . . .	44
6.13	Comparison between <i>random subset</i> and <i>uniform</i> initialization schemes of $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse and noisy data points . . . . .	45
6.14	Initial solution determined through the <i>slope-based</i> control points initialization scheme, for a partial circle derived from 60 uniformly spaced and not noisy data points . . . . .	47
6.15	<i>Slope-based</i> initial solution of $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse but not noisy data points . . . . .	48

6.16	Comparison between <i>random subset</i> and <i>slope-based</i> initialization schemes of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse but not noisy data points . . . . .	49
6.17	<i>Slope-based</i> initial solution of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points . . . . .	49
6.18	Initial solution determined through the <i>changeDir</i> control points initialization scheme, for a partial circle derived from 60 uniformly spaced and not noisy data points . . . . .	52
6.19	<i>changeDir</i> initial solution of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse but not noisy data points . . . . .	53
6.20	Comparison between <i>random subset</i> , <i>slope-based</i> and <i>changeDir</i> initialization schemes of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse but not noisy data points . . . . .	54
6.21	<i>changeDir</i> initial solution of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points . . . . .	54
6.22	Initial solution determined through the <i>changeDir</i> control points initialization scheme of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points, fixing $n = 9$ and adjusting accordingly the threshold $\gamma$ . . . . .	55
6.23	Example of noise effect on direction change of two adjacent data points: the change of direction caused by noise decreases when points are more distant . . . . .	56
6.24	$\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points . . . . .	57
6.25	$\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points, with <i>sampling</i> . . . . .	58
6.26	$\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points, with <i>sampling</i> and <i>smoothing</i> . . . . .	59
6.27	<i>changeDir</i> initial solution of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points, with <i>sampling</i> and <i>smoothing</i> . . . . .	59

6.28	Comparison between <i>random subset</i> and <i>changeDir</i> initialization schemes of $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse and noisy data points . . . . .	60
6.29	B-spline fitting curve of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points, adopting the <i>changeDir</i> initial solution with <i>sampling</i> and <i>smoothing</i> , after 2000 optimization steps . . . . .	60
6.30	Histogram of $E_{gof}$ frequencies for <i>random</i> initialization scheme obtained after fitting the $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse data points, considering 2000 optimization steps. We have 200 runs of the optimization process. . . . .	61
6.31	Histogram of $E_{gof}$ frequencies for <i>random subset</i> initialization scheme obtained after fitting the $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse data points, considering 2000 optimization steps. We have 200 runs of the optimization process. . . . .	62
6.32	Histogram of $E_{gof}$ frequencies for <i>changeDir</i> initialization scheme obtained after fitting the $\sin(x)$ curve in interval $[0, 8\pi]$ , derived from 200 sparse data points, considering 2000 optimization steps. We have 200 runs of the optimization process. . . . .	63
7.1	Comparison between <i>centripetal</i> and <i>global minimum distance</i> parametrization schemes, of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points . . . . .	68
7.2	Comparison between <i>centripetal</i> , <i>global minimum distance</i> and <i>local minimum distance</i> parametrization schemes, of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points . . . . .	70
7.3	Comparison of $E_{gof}$ in the first 8000 optimization steps for the B-spline fitting curve of the $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points, with noise level of $\alpha = 0.5$ . The initial solution is chosen with the <i>random</i> scheme	71

7.4	Comparison between <i>centripetal</i> , <i>global minimum distance</i> and their hybrid combination, of $\sin(x)$ curve in interval $[0, 8\pi]$ derived from 200 sparse and noisy data points . . . . .	72
A.1	B-spline fitting curve of self intersecting curve . . . . .	76
A.2	B-spline fitting curve of spiral shape . . . . .	77
A.3	B-spline fitting curve of hand shape . . . . .	77
A.4	B-spline fitting curve of Italy shape . . . . .	78
A.5	B-spline fitting curve of sinusoid curve . . . . .	78
A.6	B-spline fitting curve of self intersecting curve . . . . .	79
A.7	B-spline fitting curve of Trentino shape . . . . .	79

# Bibliography

- [1] EM Arkin, LP Chew, DP Huttenlocher, K. Kedem, and JSB Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.
- [2] R. Battiti, M. Brunato, and F. Mascia. Reactive search and intelligent optimization. 2007.
- [3] Roberto Battiti and Giampietro Tecchiolli. Learning with first, second and no derivatives: A case study in high energy physics. *Neurocomp*, 6:181–206, 1994.
- [4] GEP Box and M.E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, pages 610–611, 1958.
- [5] M. Brunato and R. Battiti. The reactive affine shaker: a building block for minimizing functions of continuous variables. Technical report, Technical Report DIT-06-012, Universita di Trento, 2006.
- [6] M. Brunato and R. Battiti. RASH: A Self-adaptive Random Search Method. *Adaptive and Multilevel Metaheuristics*, page 95, 2008.
- [7] Roberto Brunelli and Giampietro Tecchiolli. On random minimization of functions. *Biological Cybernetics*, 65(6):501 – 506, 1991.
- [8] C. De Boor. *A practical guide to splines*. Springer Verlag, 2001.

- [9] M. Jacob, T. Blu, and M. Unser. Efficient energies and algorithms for parametric snakes. *IEEE transactions on image processing*, 13(9):1231–1244, 2004.
- [10] S. Kabus, T. Netsch, B. Fischer, and J. Modersitzki. B-spline registration of 3D images with Levenberg-Marquardt optimization. In *Proceedings of the SPIE*, volume 5370, pages 304–313. Citeseer, 2004.
- [11] ETY Lee. Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, 21(6):363–370, 1989.
- [12] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.
- [13] H. Pottmann and M. Hofer. Geometry of the squared distance function to curves and surfaces. *Visualization and Mathematics III*, pages 223–244, 2003.
- [14] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In *Proceedings of Pacific Graphics*, volume 2, pages 8–25, 2002.
- [15] WH Press, SA Teukolsky, WT Vetterling, and BP Flannery. Numerical recipes in C++. 2002. *Cambridge UP*.
- [16] E. S. ”uli and D.F. Mayers. *An introduction to numerical analysis*. Cambridge University Press, 2003.
- [17] M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal processing magazine*, 16(6):22–38, 1999.
- [18] H. Yang, W. Wang, and J. Sun. Control point adjustment for B-spline curve approximation. *Computer-Aided Design*, 36(7):639–652, 2004.