Instytut Informatyki
Wydział Matematyki i Informatyki
Uniwersytet Wrocławski

PRZEMYSŁAW GOSPODARCZYK

OBNIŻANIE STOPNIA I SCALANIE KRZYWYCH BÉZIERA

Praca doktorska

Promotor: dr hab. Paweł Woźny

Wrocław 2016

Institute of Computer Science
Department of Mathematics and Computer Science
University of Wrocław

PRZEMYSŁAW GOSPODARCZYK

DEGREE REDUCTION AND MERGING OF BÉZIER CURVES

Ph.D. Thesis

Supervisor: dr hab. Paweł Woźny

Wrocław 2016

Acknowledgments

I would like to thank dr hab. Paweł Woźny and prof. Stanisław Lewanowicz for their valuable comments and suggestions concerning this thesis.

Wrocław, February 2016

Streszczenie

W niniejszej pracy przedstawiono różne podejścia do problemów optymalnego, w sensie normy średniokwadratowej, obniżania stopnia i scalania krzywych Béziera. Oba problemy związane są z systemami projektowania wspomaganego komputerowo. W wypadku większości systemów tego typu istnieje górne ograniczenie na stopień krzywych, które mogą być tam przetwarzane. Wspomniane ograniczenia zależą od konkretnego systemu i biorąc pod uwagę dużą ich liczbę mogą różnic się one w znaczący sposób. W związku z tym, w celu wymiany danych pomiędzy systemami, konieczna jest konwersja, którą zazwyczaj można wykonać jedynie w sposób przybliżony. Dwie główne operacje tego typu to właśnie obniżanie stopnia i scalanie krzywych.

Obniżanie stopnia krzywych Béziera polega na zastąpieniu oryginalnej krzywej Béziera stopnia n, inną krzywą Béziera niższego stopnia m. Ponadto zwykle wymaga się, aby szukana krzywa spełniała pewne dodatkowe ograniczenia na końcach przedziału parametryzacji. Najczęściej są to warunki ciągłości parametrycznej lub ich uogólnienie tj. warunki ciągłości geometrycznej. Takie podejście do problemu nazywamy konwencjonalnym. Zaprezentowano algorytmy konwencjonalnego obniżania stopnia krzywych Béziera z warunkami ciągłości geometrycznej. Oprócz tego zaproponowano nowe podejście do problemu obniżania stopnia planarnych krzywych Béziera z warunkami ciągłości parametrycznej. Po raz pierwszy nałożono tzw. ograniczenia obszaru zmienności punktów kontrolnych, co przyczyniło się do otrzymywania krzywych, których punkty kontrolne rozmieszczone są w sposób bardziej intuicyjny. Ten pomysł znacznie ułatwia dalsze modelowanie.

Scalanie krzywych Béziera polega na zastąpieniu dowolnej liczby sąsiadujących krzywych Béziera pojedynczą krzywą Béziera określonego stopnia. Dodatkowo nakłada się podobne ograniczenia jak w wypadku zadania obniżania stopnia. Podano algorytmy konwencjonalnego scalania krzywych Béziera z warunkami ciągłości parametrycznej i geometrycznej. Ponadto zaproponowano nowatorskie podejście do problemu scalania planarnych krzywych Béziera z warunkami ciągłości parametrycznej. Podobnie jak w wypadku obniżania stopnia, pokazano że nałożenie ograniczeń obszaru zmienności punktów kontrolnych pozwala uzyskać krzywe będące bardziej użyteczne w praktyce.

W wypadku podejść konwencjonalnych wykorzystano pewne własności tzw. dualnych wielomianów Bernsteina z ograniczeniami. W efekcie każda z zaprezentowanych metod ma najniższą złożoność obliczeniową spośród wszystkich tego typu metod.

Ograniczenia obszaru zmienności punktów kontrolnych powodują, że oba problemy znacznie trudniej rozwiązać. W tym celu można wykorzystać pewną metodę iteracyjną, którą da się znacząco przyspieszyć stosując algorytmy szybkiej konstrukcji i modyfikacji baz dualnych.

Rozdział 1 zawiera wstępne informacje. W rozdziale 2 pokazano związek pomiędzy warunkami ciągłości a szukanymi krzywymi. Pojęcie i własności baz dualnych zaprezentowano w rozdziale 3. W rozdziale 4 rozwiązano zadanie obniżania stopnia krzywych Béziera z warunkami ciągłości geometrycznej. Następnie sformułowano i rozwiązano problem obniżania stopnia planarnych krzywych Béziera z warunkami ciągłości parametrycznej i ograniczeniami obszaru zmienności punktów kontrolnych (zob. rozdział 5). W rozdziałach 6 i 7 można znaleźć metody scalania krzywych Béziera odpowiednio z warunkami ciągłości parametrycznej i geometrycznej. W rozdziale 8 sformułowano i rozwiązano problem scalania planarnych krzywych Béziera z warunkami ciągłości parametrycznej i ograniczeniami obszaru zmienności punktów kontrolnych.

Abstract

In this thesis, various approaches to the problems of optimal degree reduction and merging of Bézier curves with respect to the least squares norm are presented. Both problems are associated with computer aided design systems. In most cases, a system is able to process curves of upper-bounded degree. Maximum permissible degree depends on a system. Taking into account that there are many different systems, those degrees can vary quite significantly. As a consequence, the exchange of geometric data between systems often requires approximate conversion. There are two main operations that should be considered: degree reduction and merging.

Degree reduction of Bézier curves is to replace an original Bézier curve of degree n with a different Bézier curve of lower degree m. Moreover, it is often required that the searched curve satisfies some additional constraints at the endpoints. Those constraints are usually parametric continuity conditions or their generalization, namely geometric continuity conditions. Such an approach to the problem is called conventional. The algorithms of conventional degree reduction of Bézier curves with geometric continuity conditions are presented. Furthermore, a new approach to the problem of degree reduction of planar Bézier curves with parametric continuity conditions is proposed. For the first time, the so-called box constraints are imposed, which results in curves with more intuitive location of control points. This idea makes further modeling much easier.

Merging of Bézier curves is to replace an arbitrary number of adjacent Bézier curves with a single Bézier curve of certain degree. Additionally, one may impose similar constraints as in the case of the degree reduction problem. The algorithms of conventional merging of Bézier curves with parametric and geometric continuity conditions are given. Moreover, a novel approach to the problem of merging of planar Bézier curves with parametric continuity conditions is proposed. As in the case of degree reduction, it is shown that the imposition of box constraints results in curves which are practically more useful.

In the case of the conventional approaches, certain properties of the so-called constrained dual Bernstein polynomials are used. Consequently, each presented method has the lowest computational complexity among all existing methods.

The box constraints make both problems much more difficult to solve. To deal with them, one may use a certain iterative method, which can be significantly improved using fast algorithms of construction and modification of dual bases.

Chapter 1 has a preliminary character. In Chapter 2, a relation between continuity conditions and searched curves is shown. The concept and properties of dual bases are presented in Chapter 3. In Chapter 4, the problem of degree reduction of Bézier curves with geometric continuity conditions is solved. Next, the problem of degree reduction of planar Bézier curves with parametric continuity conditions and box constraints is formulated and solved (see Chapter 5). In Chapters 6 and 7, one can find methods of merging of Bézier curves with parametric and geometric continuity conditions, respectively. Finally, the problem of merging of planar Bézier curves with parametric continuity conditions and box constraints is formulated and solved (see Chapter 8).

Contents

Chapter 1

Introduction

1.1 Computer aided design

According to [58], "*Computer aided design* (*CAD*) can be defined as the use of computer systems to assist in the creation, modification, analysis or optimization of a design." *CAD system* consists of hardware and software. *CAD hardware* is usually a central processing unit with multiple work stations and some input and output devices, e.g., monitors, keyboards, mouses, printers, plotters and drafting equipment. *CAD software* is a computer program which implements computer graphics and provides some necessary tools. A set of tools depends on specific needs of a designer. CAD systems are helpful in creating precision drawings and technical illustrations, therefore, they are useful for people of various professions, including architects, engineers, drafters and artists. As a consequence, there are many types of modeling systems for different applications, e.g., *electronic design automation* (*EDA* or *ECAD*), *mechanical design automation* (*MDA*) and *computer aided drafting*.

In 1957, Patrick Hanratty, known as *the Father of CAD*, developed the first numerical control manufacturing system called *Program for Numerical Tooling Operations* (*PRONTO*). However, it is assumed that the ancestor of modern CAD systems is *Sketchpad* which was developed by Ivan Sutherland in 1963 as a part of his Ph.D. thesis at MIT. It was the first system ever to provide a complete graphical user interface. In addition, a user was able to interact graphically with the system by drawing on a monitor with a light pen (see Figure 1). In the late 1960s, large automotive and aerospace manufacturers created their own CAD systems. Those systems were developed by internal groups in association with university scientists. In the 1970s, due to fast development of computer industry and some major advances in CAD software, CAD industry became a billion dollar hardware and software business. In the 1980s, a decrease of prices of computers and maintenance costs made CAD systems available to more users. For extensive history of CAD industry, see [98].

Nowadays, companies use CAD systems to design almost every product on the market in the world. Those modern systems are based on *interactive computer graphics* (*ICG*), i.e., an implementation of a *user-friendly interface* which displays *interactive images*. The images consist of some basic objects, such as points, lines, curves, etc. To obtain a satisfying result, a designer modifies the image using tools which are provided by the system. The basic operations are scaling, translation, rotation and several other geometric transformations. Additionally, some specialized methods required by a company should be available.

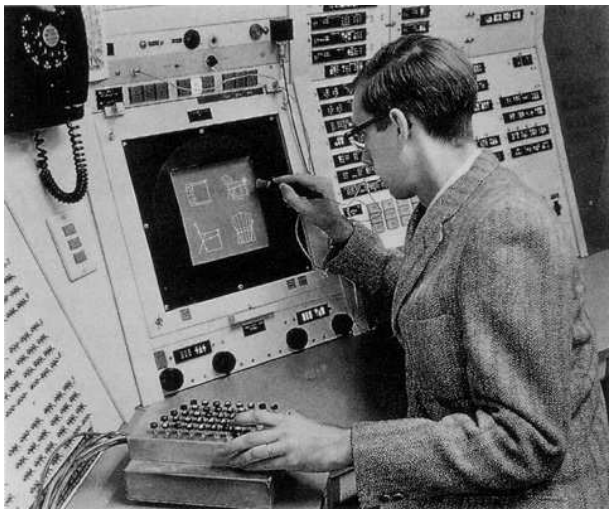


Figure 1: Ivan Sutherland's Sketchpad console. (The figure was taken from [81].)

There are several reasons why the usage of CAD systems is profitable. First of all, those systems increase efficiency of designers' work. This is achieved by detailed visualization of a designed object and by providing an efficient computing environment. Some tasks which are now considered to be trivial, used to be very time consuming before CAD systems were invented. Secondly, modeling systems improve the quality of a design by allowing to experiment and investigate many different alternative solutions in a short time. Moreover, CAD systems are very precise, therefore, number of possible errors is limited. In addition, drawings made with such systems satisfy certain standardization criteria. This is helpful in making a documentation of a design. More details on CAD systems can be found in [47, 58]. The progress of CAD systems would not be possible if not for some crucial advances in the area of curves and surfaces. In the next subsection, we look into a discipline which specializes in the development of *mathematical methods for curves and surfaces*.

1.2 Computer aided geometric design

According to [33], "*Computer aided geometric design* (*CAGD*) is a discipline dealing with computational aspects of geometric objects." A need of application of mathematical methods to certain geometric shapes arises in the areas of CAD, robotics and scientific visualization. CAGD makes use of geometry, computer graphics, numerical analysis, approximation theory, data structures and computer algebra. Moreover, there are some disciplines which are closely related to CAGD, e.g., computational geometry, geometric modeling and data fitting.

The main focus of CAGD is the manipulation of *curves* and *surfaces*. Those are usually given by sets of points and have different parametric representations using polynomial functions, piecewise polynomial functions, rational functions and piecewise rational functions. The basic objects of CAGD are *Bézier curves and surfaces*, *rational Bézier curves and surfaces*, *splines*, *B-spline curves and surfaces* and *Non-Uniform Rational B-Spline (NURBS) curves and surfaces* (see, e.g., [33, 85]). In this thesis, we are dealing with some important problems concerning Bézier curves.

In 1944, Roy Liming wrote an innovative book [67], where he transformed the classical

drafting methods into certain computational techniques. During World War II, he worked for North American Aviation (NAA). NAA used his work to build fighter planes. This was not the first use of curves in aircraft industry, however, for the first time one could express blueprints by numbers. Liming's ideas have significantly limited a number of possible mistakes caused by misinterpretations of drawings. In the 1950s, his approach was widely used by U.S. aircraft companies. Another significant contribution came from the automotive industry. In the early 1960s, Paul de Casteljau, who worked for Citroën, and Pierre Bézier, who worked for Rénault, developed independently Bézier curves and surfaces (see the pioneering papers [9–11, 24, 25] and Figure 2). Further on in this chapter, we explain why this is considered to be a turning point in CAGD. The term CAGD was used for the first time in 1974, by Robert Barnhill and Richard Riesenfeld, during the conference at the University of Utah (for the proceedings of this conference, see [5]). After this event, CAGD became a stand-alone discipline. Since powerful computers have become accessible, there has been a fast progress in this area. More detailed history of CAGD can be found in [34, §1]. See also [33, Preface and §1].

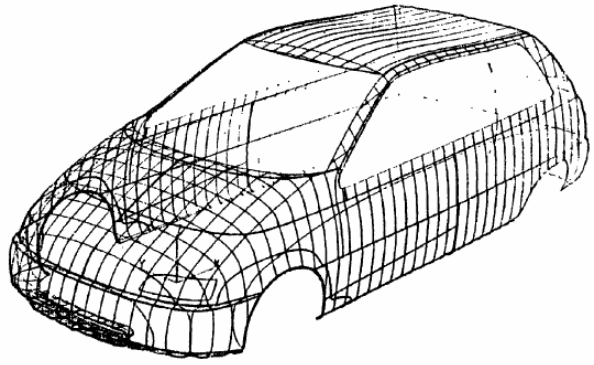


Figure 2: Cross sections of a car body. (The figure was taken from [12].)

More information about CAGD can be found in [33, 34].

In §§1.3–1.5, we focus our attention on terminology and some relevant facts about *Bernstein polynomials*, Bézier curves and *composite Bézier curves*.

1.3 Bernstein polynomials

In this subsection, we give the definition and some useful properties of Bernstein polynomials. The features stated below are meaningful, since Bézier curves are expressed in terms of these polynomials.

Bernstein polynomials of degree n are defined by

$$B_k^n(t) := \binom{n}{k} t^k (1-t)^{n-k} \qquad (k = 0, 1, \ldots, n), \tag{1.1}$$

where $\binom{n}{k} := \frac{n!}{k!(n-k)!}$ is a *binomial coefficient*. For convenience, we assume that $B_k^n(t) \equiv 0$ if $k < 0$ or $k > n$. In the thesis, we limit ourselves to the interval $t \in [0, 1]$ in which these polynomials are useful from a practical point of view. See Figure 3.

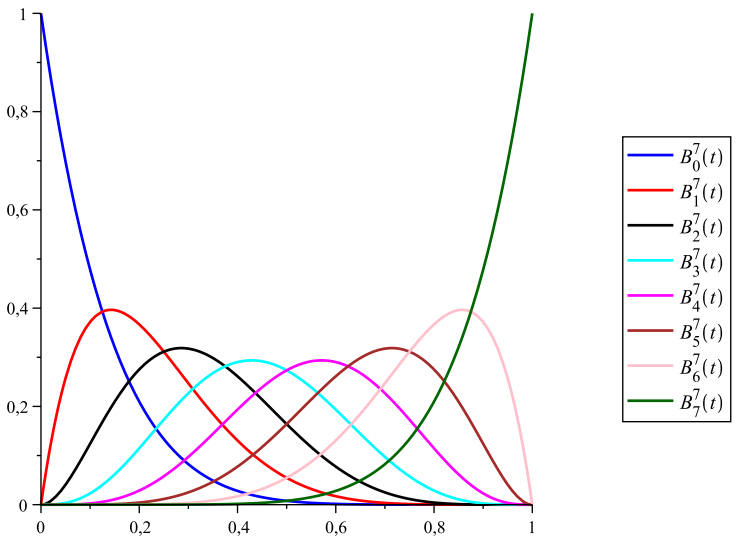


Figure 3: Bernstein polynomials of degree 7 restricted to the interval [0, 1].

Now, we give a brief description of some basic properties of these polynomials.

1. Bernstein polynomials are nonnegative over the interval [0, 1].

2. Notice that $B_k^n(0) = \delta_{k0}$ and $B_k^n(1) = \delta_{kn}$, where δ_{ij}, known as *Kronecker delta*, equals 1 if $i = j$, and 0 otherwise. Moreover, assuming that $k \neq 0$, $B_k^n(t)$ has a root with multiplicity k at $t = 0$. Similarly, when $k \neq n$, $B_k^n(t)$ has a root with multiplicity $n - k$ at $t = 1$.

3. Bernstein polynomials form a *partition of unity*,

$$\sum_{k=0}^{n} B_k^n(t) = 1.$$

A simple proof is based on the *binomial theorem* (see, e.g., [33, §5.1]).

4. There is a *symmetry* $B_{n-k}^n(1-t) \equiv B_k^n(t)$, which follows immediately from an elementary identity of binomial coefficients, namely $\binom{n}{k} = \binom{n}{n-k}$.

5. It is well known that the product of two Bernstein polynomials can be written as

$$B_k^n(t) B_l^m(t) = \frac{\binom{n}{k}\binom{m}{l}}{\binom{n+m}{k+l}} B_{k+l}^{n+m}(t) \qquad (k = 0, 1, \ldots, n;\ l = 0, 1, \ldots, m).$$

6. On the interval [0, 1], $B_k^n(t)$ $(n \neq 0)$ has a unique local maximum at $t = k/n$.

7. Let Π_n denote the space of all polynomials of degree at most n. It is well known that the Bernstein polynomials $B_0^n, B_1^n, \ldots, B_n^n$ form a *basis* of this space (for the proof, see [8, §10.2]).

8. Bernstein and *monomial* bases are related in the following way:

$$t^k = \sum_{i=k}^{n} \frac{\binom{i}{k}}{\binom{n}{k}} B_i^n(t), \quad B_k^n(t) = \sum_{i=k}^{n} (-1)^{i-k} \binom{n}{i}\binom{i}{k} t^i \qquad (k = 0, 1, \ldots, n).$$

The proofs are given in [42, §5.5.1]. In particular, for $k = 0$, the first formula becomes a partition of unity (see, pt. 3). Notice that these identities can be used to prove the previous property.

9. The following recurrence relation holds (for the proof, see [33, §5.1]):

$$B_k^n(t) = (1-t)B_k^{n-1}(t) + tB_{k-1}^{n-1}(t) \qquad (k = 0, 1, \ldots, n). \tag{1.2}$$

10. The *subdivision formula* is as follows:

$$B_k^n(ct) = \sum_{i=k}^{n} B_i^n(t) B_k^i(c) \qquad (k = 0, 1, \ldots, n).$$

To prove this one, we apply the binomial expansion to the left-hand side of the equation. Then, some simple calculations lead to the result.

11. The derivative of B_k^n can be represented as

$$\frac{\mathrm{d}}{\mathrm{d}t} B_k^n(t) = n \left[B_{k-1}^{n-1}(t) - B_k^{n-1}(t) \right], \tag{1.3}$$

which is fairly easy to verify. The rth derivative of B_k^n is given by the following formula:

$$\frac{\mathrm{d}^r}{\mathrm{d}t^r} B_k^n(t) = \frac{n!}{(n-r)!} \sum_{i=\max(0,k+r-n)}^{\min(k,r)} (-1)^{i+r} \binom{r}{i} B_{k-i}^{n-r}(t) \qquad (r = 1, 2, \ldots).$$

See [28, Theorem 3.1], for the inductive proof of this identity.

12. The integrals of B_k^n can be expressed as

$$\int_0^u B_k^n(t)\,\mathrm{d}t = \frac{1}{n+1} \sum_{i=k+1}^{n+1} B_i^{n+1}(u), \qquad \int_0^1 B_k^n(t)\,\mathrm{d}t = \frac{1}{n+1}.$$

The first formula, which obviously yields the second one, can be proven using the technique of integration by parts.

13. There are three *degree elevation formulas*,

$$tB_k^n(t) = \frac{k+1}{n+1} B_{k+1}^{n+1}(t), \qquad (1-t)B_k^n(t) = \left(1 - \frac{k}{n+1}\right) B_k^{n+1}(t),$$

$$B_k^n(t) = \left(1 - \frac{k}{n+1}\right) B_k^{n+1}(t) + \frac{k+1}{n+1} B_{k+1}^{n+1}(t),$$

for $k = 0, 1, \ldots, n$. The first two of these, which clearly imply the third one, are evident.

As we shall see in the next subsection, the above-mentioned facts have a key impact on the attributes of Bézier curves. For further properties of Bernstein polynomials, see [36, §5.1].

Bernstein polynomials were introduced by Sergei Natanovich Bernstein in 1912. His goal was to prove the *Weierstrass approximation theorem*, i.e., to show that one can use polynomials to approximate every continuous function, with a chosen precision, over any closed and bounded interval. Bernstein provided an explicit converging sequence of polynomials, therefore, the proof is constructive (see, e.g., [23, §10.3]). However, the convergence is very slow. For that reason, significance of the proof is only theoretical. As it turned out in the early 1960s, Bernstein polynomials have found their more practical application in the area of curves and surfaces. In the next subsection, we explain this application. More information on the history of Bernstein polynomials is available in [36, §§1–4]. Some other applications of these polynomials are given in [36, §9].

1.4 Bézier curves

We start this subsection with the definition of a Bézier curve.

Let Π_n^d denote the space of all parametric polynomials in $\mathbb{R}^d$ of degree at most n; $\Pi_n^1 \equiv \Pi_n$ (cf. §1.3, pt. 7). A Bézier curve $P \in \Pi_n^d$ is the following parametric curve:

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t) \qquad (t \in [0,\, 1]), \tag{1.4}$$

where $p_0, p_1, \ldots, p_n \in \mathbb{R}^d$ are called *control points* (or *Bézier points*), n is the *degree* of the curve, and $B_0^n, B_1^n, \ldots, B_n^n$ are Bernstein polynomials of degree n given by (1.1).

Example 1.1. Consider the following example of a cubic ($n = 3$) Bézier curve in $\mathbb{R}^2$, defined by the control points $(0, 0)$, $(0.2, 0.95)$, $(0.66, 1)$, $(1, 0.15)$. The polygon formed by connecting the consecutive control points with lines is called *control polygon* (or *Bézier polygon*). See Figure 4.

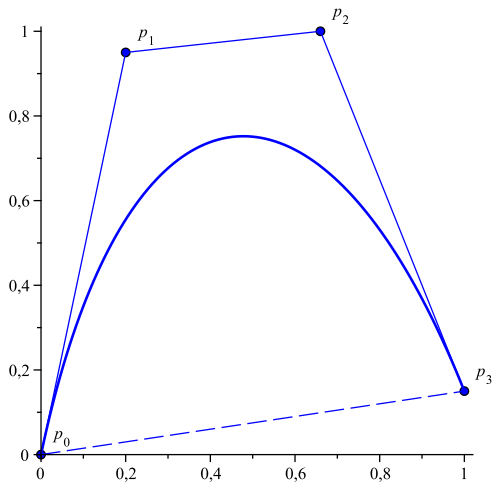


Figure 4: A simple cubic Bézier curve with its control points and the corresponding control polygon.

In the early 1960s, the intention of Paul de Casteljau and Pierre Bézier was to create some new *computer-implemented techniques* which would allow for intuitive design of automobile bodies. Such *free-form shapes* could not be described with any basic geometric quantities. Therefore, they had to find some other way to represent them with finite data. A polynomial curve seemed to be the right choice, however, appropriate basis had to be selected. Then, the curve could be represented with the coefficients of the linear combination. Selection of a proper basis was crucial. The most obvious choice, i.e., the monomial basis, is completely unsuitable, mainly due to its lack of insight into a geometrical behavior of the curve. In addition, *numerical properties* of some fundamental methods associated with this basis are poor (see, e.g., [85, §1.3]). Bernstein polynomial basis, on the other hand, is much better in this regard (see [22, 35, 37, 38]). Further on in this subsection, we give some arguments to support this statement. More facts about the work of Paul de Casteljau and Pierre Bézier can be found in [9–11], [12, §1], [24–26], [33, §1] and [36, §4].

Now, we briefly characterize some fundamental properties of Bézier curves.

1. A Bézier curve passes through its first and last control points, i.e., $P(0) = p_0$ and $P(1) = p_n$. Taking into account the well-known properties of Bernstein polynomials (see §1.3, pt. 2), the above-mentioned fact, known as the *endpoint interpolation property*, is obvious. In general, a Bézier curve does not interpolate the remaining control points.

2. A Bézier curve lies in the *convex hull* of its control points. This so-called *convex hull property* follows from §1.3, pts. 1 and 3.

3. Given the fact that *barycentric combinations* are invariant under *affine transformations*, the property §1.3, pt. 3 implies the *affine invariance* of a Bézier curve.

4. We are able to define a Bézier curve over $t \in [a, b]$, using the formula

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(u) \qquad (u := (t-a)/(b-a)).$$

This property is called *invariance under affine parameter transformations*.

5. Remembering the symmetry property of Bernstein polynomials (see §1.3, pt. 4), we observe that

$$\sum_{i=0}^{n} p_i B_i^n(t) = \sum_{i=0}^{n} p_{n-i} B_i^n(1-t)$$

and conclude that the control points $p_0, p_1, \ldots, p_n$ and $p_n, p_{n-1}, \ldots, p_0$ represent the same Bézier curve. However, note that the *traversal directions* are different.

6. The derivative of nth degree Bézier curve P is the following vector:

$$P'(t) = n \sum_{i=0}^{n-1} (p_{i+1} - p_i)\, B_i^{n-1}(t). \tag{1.5}$$

Clearly, this identity can be easily proven using (1.3) (see, e.g., [85, §1.3]). Higher order derivatives of P can be obtained by repeated application of the formula (1.5). As a result, one can prove by induction that the rth derivative of P can be written as

$$P^{(r)}(t) = \frac{n!}{(n-r)!} \sum_{i=0}^{n-r} \Delta^r p_i B_i^{n-r}(t) \qquad (r = 1, 2, \ldots),$$

where Δ is the *forward difference operator* defined recursively,

$$\Delta^0 p_i := p_i, \qquad \Delta^r p_i := \Delta^{r-1} p_{i+1} - \Delta^{r-1} p_i \qquad (r = 1, 2, \ldots), \tag{1.6}$$

and explicitly (see, e.g., [82, Lemma 4.1]),

$$\Delta^r p_i := \sum_{j=0}^{r} (-1)^{r+j} \binom{r}{j} p_{i+j} \qquad (r = 0, 1, \ldots). \tag{1.7}$$

Notice that for $t = 0, 1$, the following formulas hold:

$$P^{(r)}(0) = \frac{n!}{(n-r)!} \Delta^r p_0, \qquad P^{(r)}(1) = \frac{n!}{(n-r)!} \Delta^r p_{n-r}. \tag{1.8}$$

As we shall see later, these cases are particularly important.

7. It is possible to express a Bézier curve of degree n in the Bernstein basis of degree $n+1$. To do so, we recall the degree elevation formulas (see §1.3, pt. 13) and apply some simple manipulations (see, e.g., [33, §6.1]). As a result, we get

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t) = \sum_{i=0}^{n+1} q_i^{(1)} B_i^{n+1}(t),$$

where

$$q_i^{(1)} := \frac{i}{n+1} p_{i-1} + \left(1 - \frac{i}{n+1}\right) p_i \qquad (i = 1, 2, \ldots, n), \tag{1.9}$$

$q_0^{(1)} := p_0$ and $q_{n+1}^{(1)} := p_n$. In order to find representation in the Bernstein basis of degree $n+r$, the *unit degree elevation formula* can be applied repeatedly. However, there is a well-known explicit formula,

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t) = \sum_{i=0}^{n+r} q_i^{(r)} B_i^{n+r}(t),$$

where

$$q_i^{(r)} := \sum_{j=\max\{0, i-r\}}^{\min\{n, i\}} p_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}} \qquad (i = 0, 1, \ldots, n+r).$$

This *r-fold degree elevation formula* can be proven by induction.

8. Let $P(t) \equiv P_n(p_0, p_1, \ldots, p_n)$. The identity (1.2) leads to the following recurrence relation:

$$P_n(p_0, p_1, \ldots, p_n) = (1-t) P_{n-1}(p_0, p_1, \ldots, p_{n-1}) + t P_{n-1}(p_1, p_2, \ldots, p_n),$$

which yields the most fundamental algorithm associated with Bézier curves, namely *de Casteljau algorithm*. Assume that t_0 is fixed, then $P(t_0)$ can be evaluated using Algorithm 1.2.

Algorithm 1.2 ([24, 25]). *[de Casteljau algorithm]*
Input: value of parameter $t_0 \in [0, 1]$, *control points* $p_i \equiv p_i^{(0)}$ $(i = 0, 1, \ldots, n)$
Output: point on a curve, i.e., $P(t_0) \equiv p_0^{(n)}$

`Step 1.` *For* $j = 1, 2, \ldots, n$, *compute*

$$p_i^{(j)} := (1 - t_0) \cdot p_i^{(j-1)} + t_0 \cdot p_{i+1}^{(j-1)} \qquad (i = 0, 1, \ldots, n - j).$$

`Step 2.` *Return* $p_0^{(n)}$.

Clearly, the complexity of the algorithm is $O(n^2)$. This simple method has a meaningful and intuitive geometric interpretation. We subdivide each line segment of the control polygon with the ratio $t_0 : (1 - t_0)$. Next, we connect the resulting points to get a new polygon which is shorter by one segment. We repeat these operations until we get a single point. To visualize this geometric construction, let us revisit Example 1.1. We set $t_0 := 0.6$ and apply the algorithm. For the result, see Figure 5.

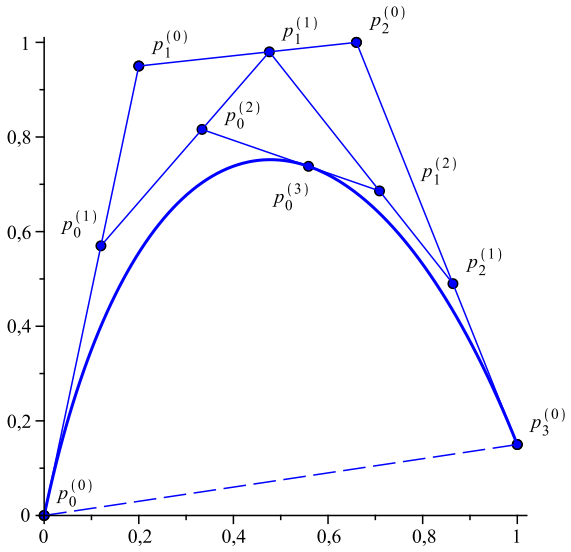


Figure 5: Geometric interpretation of de Casteljau algorithm applied to the cubic Bézier curve (see Example 1.1).

From a historical perspective, de Casteljau algorithm was the first and crucial step towards the development of Bézier curves. For extensive information on this algorithm, we recommend [12]. See also [33, §4].
To evaluate a point on a Bézier curve, one could also use a linear time algorithm related to the *Horner scheme* (also known as *Horner's method* and *Horner's rule*) which is associated with the monomial basis (see, e.g., [86, §2.3]). However, if coefficients of the monomial form vary greatly in magnitude, then this approach should be avoided because of *round-off errors* (see [22, 37, 38, 85]).

9. De Casteljau algorithm can be used to *subdivide* a Bézier curve. Suppose that we subdivide the curve (1.4) at $t = t_0$. Consequently, we obtain two nth degree Bézier curves corresponding to the intervals $[0,\, t_0]$ and $[t_0,\, 1]$, respectively. It can be proven (see, e.g., [16, §27]) that the control points of those curves are

$$p_0, p_0^{(1)}, \ldots, p_0^{(n)} \quad \text{and} \quad p_0^{(n)}, p_1^{(n-1)}, \ldots, p_n,$$

respectively.

One of the major advantages of Bézier curves is that they can be easily joined to form a more complex shape. In the next subsection, we investigate this possibility. For further properties of Bézier curves, see [33, §§4–6].

1.5 Composite Bézier curves

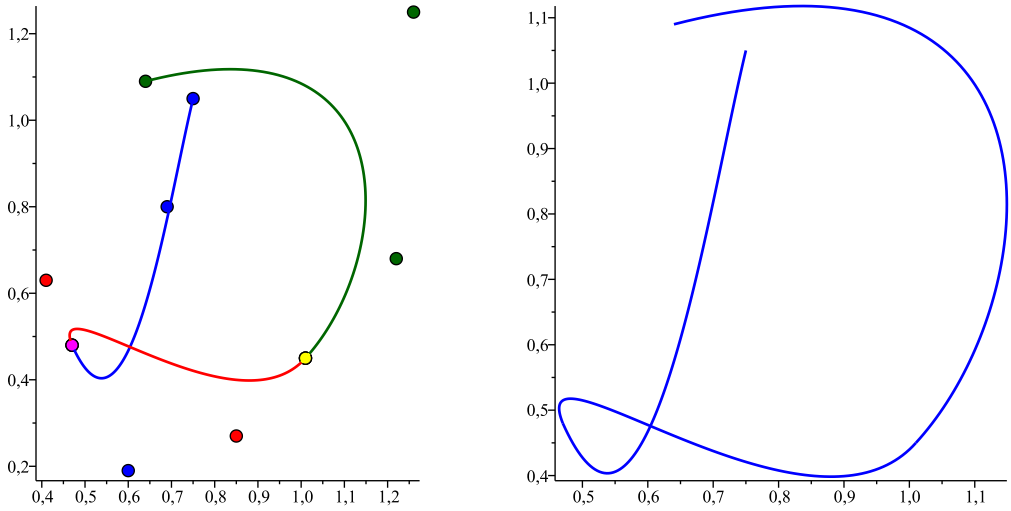
A single Bézier curve is not flexible enough to represent more complex shapes. In practice, there are two options to deal with this issue. The first strategy is to add some new control points and, therefore, increase the degree of a single curve (see §1.4, pt. 7). Unfortunately, many CAD systems have their limitations on the maximum degree of curves that can be processed. Furthermore, modeling of higher degree curves can be uncomfortable since the modification of a single control point affects the whole curve. In addition, such curves are more expensive to evaluate. Alternatively, one can construct a *piecewise parametric curve* of several segments. Each segment is a Bézier curve *smoothly* connected with the adjacent ones. Quadratic (degree 2) and cubic (degree 3) segments are the most common and the easiest to work with. However, one must pay special attention to the *continuity at the endpoints*. This might be inconvenient during the modeling process.

Now, we give the formal definition of a composite Bézier curve. Let $0 = t_0 < t_1 < \ldots < t_s = 1$ be a partition of the interval $[0,\, 1]$. A composite Bézier curve (also known as *piecewise Bézier curve* and *Bézier spline*) P is a piecewise parametric curve which in the interval $[t_{i-1},\, t_i]$ $(i = 1, 2, \ldots, s)$ is exactly represented as a Bézier curve $P^i \in \Pi^d_{n_i}$,

$$P(t) = P^i(t) := \sum_{j=0}^{n_i} p_j^i \, B_j^{n_i}\left(\frac{t - t_{i-1}}{\Delta t_{i-1}}\right) \qquad (t \in [t_{i-1},\, t_i]),$$

where $p_j^i \in \mathbb{R}^d$.

Example 1.3. We introduce the composite Bézier curve "D", formed by three cubic segments which are defined by the control points $\{(0.75, 1.05), (0.69, 0.8), (0.6, 0.19), (0.47, 0.48)\}$, $\{(0.47, 0.48), (0.41, 0.63), (0.85, 0.27), (1.01, 0.45)\}$ and $\{(1.01, 0.45), (1.22, 0.68), (1.26, 1.25), (0.64, 1.09)\}$, respectively (see Figure 6a). The partition of the interval $[0,\, 1]$ is as follows: $t_0 = 0,\ t_1 = 0.32,\ t_2 = 0.56,\ t_3 = 1$. Figure 6b illustrates the smoothness of this composite curve. A human eye is unable to notice the endpoints of the consecutive segments.



(a) (b)

Figure 6: The composite Bézier curve "D". In Figure (a), each segment and the corresponding control points are marked with a different color.

More information on composite Bézier curves can be found in [33, §5.5].

1.6 Approximate conversion of Bézier curves – overview

In most cases, a modeling system is able to process curves of degrees less or equal to a fixed maximum degree. The maximum degree depends on a system. Consequently, those degrees can vary quite significantly. Since there are many ways to represent curves, the *exchange of geometric data* between CAD systems often requires *approximate conversion*. Moreover, some operations, such as *trimming*, may result in degree elevation (see, e.g., [92, §1]). Therefore, it may happen that a CAD system is unable to handle the resulting curve and conversion is necessary. As it was stated in the pioneering paper by Hoschek [50], there are two main operations that should be considered: *degree reduction* and *merging*. Those procedures are of great importance not only because of the limitations of some modeling systems. Both of them can be applied for the sake of *data compression* and *data comparison*. In this thesis, various approaches to the problems of degree reduction and merging of Bézier curves are studied.

1.6.1 Degree reduction of Bézier curves

To begin with, we formulate the problem of *unconstrained degree reduction of Bézier curves*. This is the most basic version of the degree reduction problem.

Problem 1.4. *[Unconstrained degree reduction of Bézier curves]*
Approximate an original Bézier curve

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t)$$

with another Bézier curve

$$R(t) := \sum_{i=0}^{m} r_i B_i^m(t)$$

of lower degree ($m < n$).

First of all, one should be aware that, in general, exact degree reduction is not possible. It can only be done in the case of *degree elevated curves* (see §1.4, pt. 7). To detect the *redundancy*, one has to know the *true degree* of the original curve. This is much easier to determine if the curve is in the monomial form,

$$P(t) = \sum_{i=0}^{n} a_i t^i,$$

where

$$a_i := \sum_{j=0}^{i} (-1)^{i-j} \binom{n}{i} \binom{i}{j} p_j \qquad (i = 0, 1, \ldots, n)$$

(cf. §1.3, pt. 8). Clearly, if $a_n = a_{n-1} = \ldots = a_{n-r+1} = 0 \neq a_{n-r}$ and $m \geq n - r$, then we are able to *reverse* the process of degree elevation and get the exact result, namely the control points

$$r_i = \sum_{j=0}^{i} (-1)^{i-j} \frac{\binom{i-j+n-m-1}{n-m-1} \binom{n}{j}}{\binom{m}{i}} p_j \qquad (i = 0, 1, \ldots, m)$$

(see [36, 38]). This type of strategy can also be used in the general case (see [32, 33, 40, 54, 83–85, 95]), i.e., we pretend that P is a degree elevated curve and reverse the imagined degree elevation process. However, the results of this approach are rather poor (see Example 1.5).

In practice, a designer is dealing with large scenes formed by many Bézier curves joined end-to-end. To maintain these connections, one must impose some *continuity constraints* at the endpoints. Let us consider the most elementary restrictions, namely the *endpoint interpolation constraints*,

$$P(0) = R(0), \qquad P(1) = R(1). \tag{1.10}$$

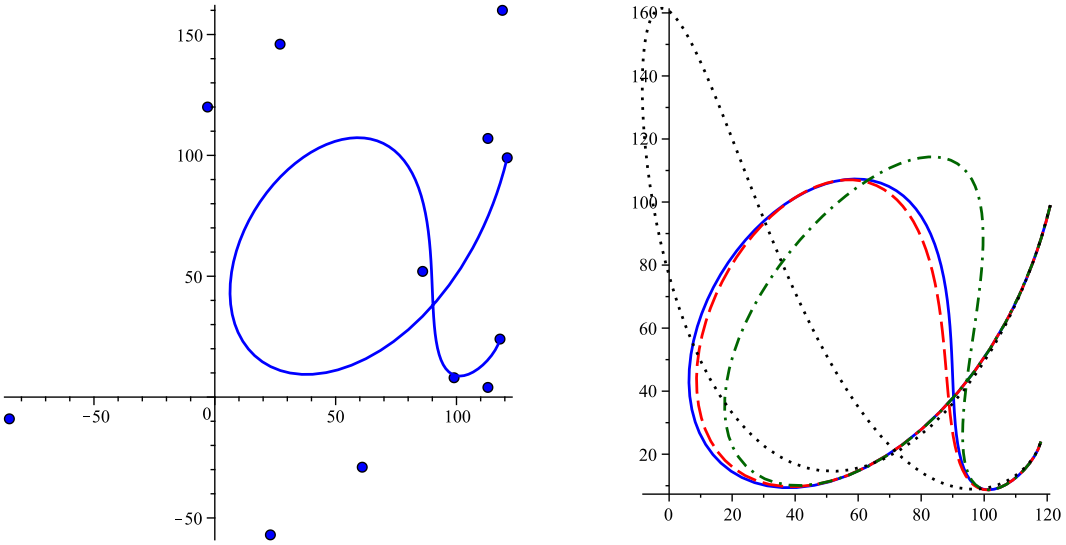
These conditions imply that $r_0 = p_0$ and $r_m = p_n$ (see §1.4, pt. 1), thus they are very easy to satisfy.

Example 1.5. Now, let us examine the algorithm of Piegl and Tiller (see [84], [85, §5.6] or [54]). For a given Bézier curve of degree n, their approach is to rewrite the degree elevation formula (1.9) and obtain a curve of degree $n - 1$. Obviously, a Bézier curve of any degree m ($m < n$) can be computed by the repeated application of the algorithm. In addition, the resulting curve always satisfies the conditions (1.10). A more detailed description of the method is given in the above-mentioned sources. For the purposes of the experiment, we consider the Bézier curve "alpha" of degree 11. The control points are given in [103,

Example 6.1]. Figure 7a illustrates the original curve with its control points. We apply the algorithm multiple times in order to obtain Bézier curves of degrees 10, 9 and 6. In each case, we compute the *maximum error*

$$E_\infty := \max_{t \in T_M} \|P(t) - R(t)\| \approx \max_{t \in [0,1]} \|P(t) - R(t)\|, \tag{1.11}$$

where $\|\cdot\|$ denotes the *Euclidean vector norm*, and $T_M := \{0, 1/M, 2/M, \ldots, 1\}$ with $M := 500$. The results are presented in Figure 7b.



(a) (b)

Figure 7: Figure (a) shows the original Bézier curve with its control points. In Figure (b), we see the results of reduction from degree 11 (blue solid line) to degrees 10 (red dashed line), 9 (green dash-dotted line) and 6 (black dotted line), using the method of Piegl and Tiller [84].

As we consider the resulting curves of different degrees, it can be seen that the only acceptable one is that of degree 10 ($E_\infty = 3.30e{+}0$). However, even in this case, the error is quite significant. Clearly, the curves of degrees 9 ($E_\infty = 2.81e{+}1$) and 6 ($E_\infty = 8.47e{+}1$) are unsatisfactory. Taking into account these poor results, algorithms of such a type are considered to be obsolete and should be avoided in contemporary CAD systems. As we shall see, there are some modern methods which lead to much better results.

In the case of Bézier curves without the redundancy, the only reasonable approach to the problem is to *minimize* a properly defined *error function*. Error of the approximation is usually measured using L_2*-norm* (*least squares approximation*),

$$\|P - R\|_{L_2} := \sqrt{\int_0^1 \|P(t) - R(t)\|^2 \, \mathrm{d}t}, \tag{1.12}$$

weighted L_2*-norm* (*weighted least squares approximation*),

$$\|P - R\|_{L_2}^{(\alpha,\beta)} := \sqrt{\int_0^1 (1-t)^\alpha t^\beta \|P(t) - R(t)\|^2 \, \mathrm{d}t} \qquad (\alpha,\, \beta > -1), \tag{1.13}$$

or L_∞-*norm* (*uniform approximation*),

$$\|P - R\|_{L_\infty} := \max_{t\in[0,\,1]} \|P(t) - R(t)\|,$$

where $\|\cdot\|$ denotes the Euclidean vector norm. Notice that $\|\cdot\|_{L_2} \equiv \|\cdot\|_{L_2}^{(0,0)}$.

Degree reduction of Bézier curves with respect to the L_∞*-norm* (see [1, 13, 15, 18, 30, 55, 57, 95, 97]) is no longer studied. The algorithms are mainly based on the properties of the so-called *constrained Chebyshev polynomials* (see, e.g., [57, §3]). In general, those polynomials cannot be represented in the explicit form, therefore, one has to implement a *Remes-type algorithm* to approximate them. Consequently, such methods are rather slow, complicated and there are no explicit formulas for an optimal solution. Alternatively, one can use the so-called *constrained Jacobi polynomials* (see, e.g., [55, §2 and §4]) to obtain explicitly a fairly good (but not optimal) approximation. In [18], one can find another explicit method of the nearly best uniform approximation, which is compared with the explicit optimal least squares approximation algorithm. As it turns out, the latter approach gives more accurate results. Further on in this thesis, the uniform approximation is omitted because of the above-mentioned drawbacks.

Degree reduction of Bézier curves with respect to the L_2*-norm* (and *weighted* L_2*-norm*) has been extensively studied (see [2, 3, 15, 18, 31, 44, 60, 68, 70, 73, 74, 76–79, 89–92, 94, 95, 103, 110–114]). Most of the algorithms solve a *system of normal equations* (see, e.g., [92]). This simple approach is *computationally expensive*, i.e., the complexity is $O(m^3)$ (cf. Problem 1.4). Furthermore, it suffers from *poor numerical properties*, since it involves *matrix inversion*. As is well known, *orthogonal polynomials* play a crucial role in the theory of the least squares approximation. However, Bernstein polynomials are not orthogonal. To overcome this difficulty, one can use certain transformations between Bernstein and orthogonal polynomial bases, e.g., *Chebyshev polynomials of the first kind* (see [89]), *Chebyshev polynomials of the second kind* (see [78]) or *Legendre polynomials* (see [60]). Unfortunately, the cost of those transformations is cubic with respect to n. Moreover, according to [78], degree reduction by the transformation matrices may be *ill-conditioned*. In [103], Woźny and Lewanowicz propose a more sophisticated approach which is based on the use of the so-called *dual Bernstein basis polynomials*. More information on those polynomials can be found in §3.3. The complexity of the method is $O(nm)$, which is significantly less than complexity of other known methods. Additionally, the algorithm avoids matrix inversion and explicit basis transformation. In [44], which is the author's joint work with Lewanowicz and Woźny, one can find a generalization of the previously mentioned approach. For a detailed description of this method, see Chapter 4.

In [17], degree reduction of Bézier curves with respect to the *Hausdorff distance*,

$$d_H(P, R) := \max\left\{\max_{p\in P}\min_{r\in R}\|p - r\|,\ \max_{r\in R}\min_{p\in P}\|p - r\|\right\}, \tag{1.14}$$

is introduced. However, it is difficult to minimize (1.14) directly. The authors of [17] use *reparametrization* of the original curve and they minimize the following L_2-distance:

$$\sqrt{\int_0^1 \|P(\varphi(t)) - R(t)\|^2\,\mathrm{d}t},$$

where $\varphi : [0,\,1] \to [0,\,1]$ is a strictly increasing continuous function such that $P(\varphi(t_0))$ is the closest point to $R(t_0)$ for all $t_0 \in [0,\,1]$. They claim that this approach produces an effect which is similar to the result of minimization of (1.14).

Example 1.6. Once again, let us consider the Bézier curve "alpha" of degree 11 (see Example 1.5). This time, we apply the algorithm of Woźny and Lewanowicz [103] in order to perform an optimal degree reduction with respect to the L_2-norm ($\alpha = \beta = 0$). As in the previous example, we impose the endpoint interpolation constraints (1.10). The results are shown in Figure 8 (cf. Figure 7b). Clearly, the resulting curves of degrees 10 ($E_\infty = 2.98e{-}2$) and 9 ($E_\infty = 1.19e{-}1$) are perfect. In contrast to the previous example, a human eye is unable to distinguish them from the original curve. As it turns out, it is also possible to obtain satisfactory curve of degree 6 ($E_\infty = 2.57e{+}0$). Compare the maximum errors with the ones from Example 1.5. Obviously, the approach of Woźny and Lewanowicz [103] is much better than that of Piegl and Tiller [84] (cf. Figure 7b).

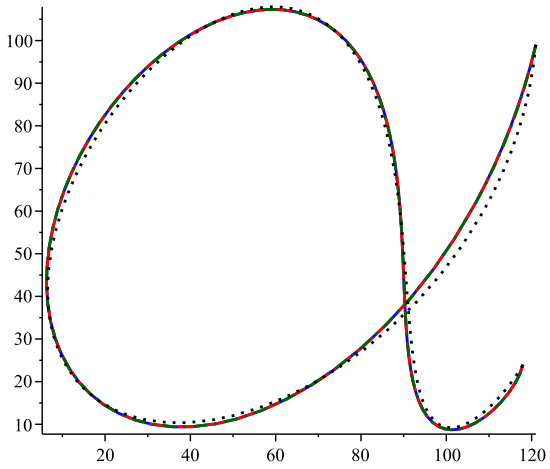


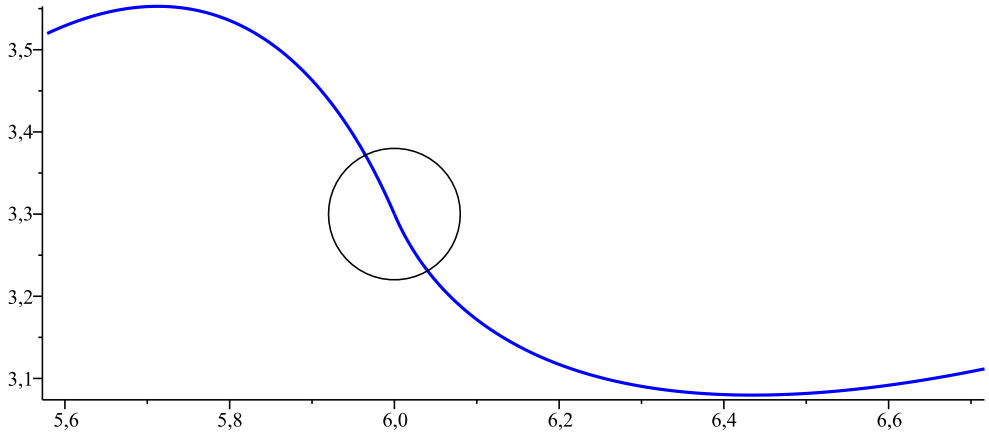
Figure 8: Optimal degree reduction with respect to the L_2-norm. The Bézier curve of degree 11 (blue solid line) reduced to the Bézier curves of the following degrees: 10 (red dashed line), 9 (green dash-dotted line) and 6 (black dotted line).

In the case of degree reduction of many Bézier curves joined end-to-end, the endpoint interpolation constraints (1.10) guarantee maintenance of the connections. However, those minimal continuity requirements do not rule out *rough edges* and *corners*. In many practical cases, a goal is to preserve *smooth connections* (see §1.5). That is to say, a human eye should not be able to notice the endpoints of the adjacent curves. In order to see the issue clearly, let us consider the following example.

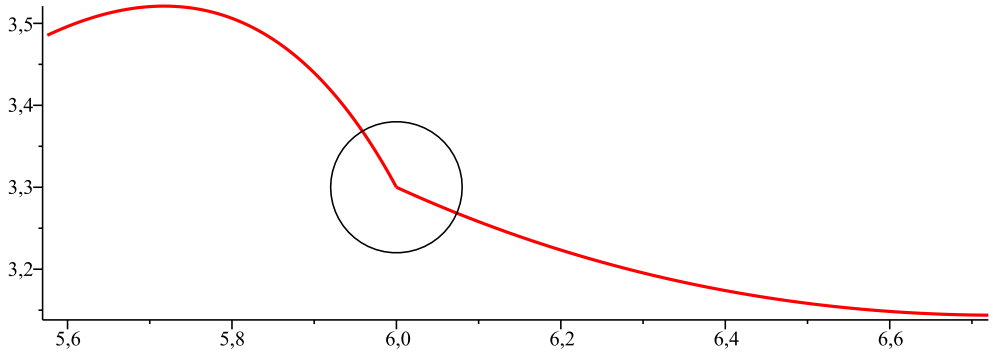
Example 1.7. Let P and Q denote two adjacent Bézier curves of degrees 5 and 6, defined by the control points $\{(2.5, 0), (3.5, 1), (4.5, 1.5), (5, 3.5), (5.7, 4), (6, 3.3)\}$ and $\{(6, 3.3), (6.23, 2.76), (7.5, 3), (8.5, 4.5), (9, 3), (10, 6), (11, 1)\}$, respectively (cf. [115, Example 2]). See Figure 9a. Clearly, these curves connect in a smooth way. Now, we apply the algorithm of Woźny and Lewanowicz [103] separately to each curve. As a result, we obtain two Bézier curves R and S of degrees 3 ($E_\infty = 7.06e{-}2$) and 4 ($E_\infty = 1.66e{-}1$), respectively. Both of them satisfy the endpoint interpolation constraints. Unfortunately, these restrictions are insufficient, i.e., the connection is not smooth enough (see Figure 9b). To avoid this defect, we must additionally preserve the *continuity of the derivatives at the endpoints*,

$$P'(1) = R'(1), \qquad Q'(0) = S'(0). \tag{1.15}$$

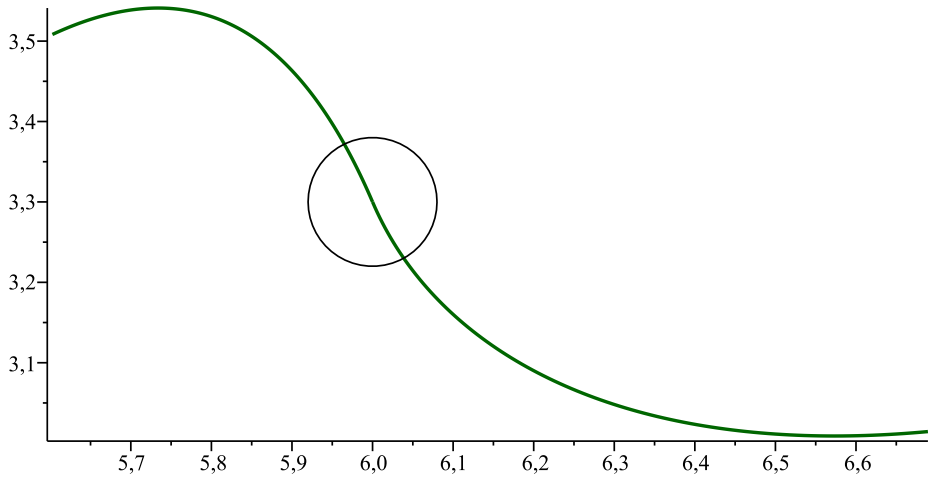
The algorithm of Woźny and Lewanowicz [103] allows us to impose such conditions. Once again, we obtain two Bézier curves of degrees 3 ($E_\infty = 8.29e{-}2$) and 4 ($E_\infty = 2.30e{-}1$). This time, the connection is smooth (see Figure 8c). However, one should realize that because of the additional restrictions, the approximation errors must be inevitably larger than for the previous results.



(a)



(b)



(c)

Figure 7: Let $t \in [0.8, 1]$ for the left curve and $t \in [0, 0.2]$ for the right one. Figure (a) shows two original Bézier curves of degrees 5 and 6. In Figure (b), we see two reduced Bézier curves of degrees 3 and 4 satisfying the endpoint interpolation constraints. The curves in Figure (c) additionally fulfill the conditions (1.15).

Example 1.7 shows the importance of continuous derivative. In fact, one can go even further and preserve the continuity of higher order derivatives. These requirements constitute the so-called *parametric continuity constraints* (also known as $C^{k,l}$ *continuity constraints*). Now, we formulate the problem of $C^{k,l}$*-constrained degree reduction of Bézier curves.*

Problem 1.8. *[$C^{k,l}$-constrained degree reduction of Bézier curves]*
For a given Bézier curve P of degree n,

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t),$$

find a Bézier curve R of lower degree m,

$$R(t) := \sum_{i=0}^{m} r_i B_i^m(t),$$

so that the following conditions are satisfied:

(i) weighted L_2-error

$$\|P - R\|_{L_2}^{(\alpha,\beta)} = \sqrt{\int_0^1 (1-t)^\alpha t^\beta \|P(t) - R(t)\|^2 \,\mathrm{d}t} \qquad (\alpha,\, \beta > -1)$$

is minimized in the space Π_m^d;

(ii) P and R are $C^{k,l}$-continuous ($k, l \geq -1$ and $k + l < m - 1$) at the endpoints, i.e.,

$$\left.\begin{aligned} R^{(i)}(0) &= P^{(i)}(0) \qquad (i = 0, 1, \ldots, k), \\ R^{(j)}(1) &= P^{(j)}(1) \qquad (j = 0, 1, \ldots, l). \end{aligned}\right\} \tag{1.16}$$

Problems of the above type are discussed in many papers [2, 3, 18, 31, 73, 78, 94, 95, 103, 110–112], usually under some simplifying assumptions, e.g., $\alpha = \beta = 0$ or $k = l$. Paper of Woźny and Lewanowicz [103] deals with Problem 1.8 in its most general form. As we have already mentioned, their approach has gained recognition because of the lowest computational complexity and some *good numerical properties*. Consequently, it may seem that further improvement in this area is impossible. However, in practice, $C^{k,l}$ constraints tend to be too restrictive. We shall see this in Chapter 4. As a result, Problem 1.8 is not as interesting as it used to be (notice the publication dates of the above-mentioned articles). For a detailed discussion on parametric continuity, see §2.1.

Now, we formulate the problem of *$G^{k,l}$-constrained degree reduction of Bézier curves* which differs from Problem 1.8 in considering, instead of the conditions (1.16), the *geometric continuity constraints* (also known as $G^{k,l}$ *continuity constraints*) at the endpoints of the curves.

Problem 1.9. *[$G^{k,l}$-constrained degree reduction of Bézier curves]*
For a given Bézier curve P of degree n,

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t), \tag{1.17}$$

find a Bézier curve R of lower degree m,

$$R(t) := \sum_{i=0}^{m} r_i B_i^m(t), \tag{1.18}$$

so that the following conditions are satisfied:

(i) weighted L_2-error

$$\|P - R\|_{L_2}^{(\alpha,\beta)} = \sqrt{\int_0^1 (1-t)^\alpha t^\beta \|P(t) - R(t)\|^2 \,\mathrm{d}t} \qquad (\alpha,\, \beta > -1) \tag{1.19}$$

is minimized in the space Π_m^d;

(ii) P and R are $G^{k,l}$-continuous ($-1 \le k,\, l \le 3$ and $k + l < m - 1$) at the endpoints, i.e.,

$$\left.\begin{aligned} R^{(i)}(t) &= P^{(i)}(\varphi(t)) \qquad (t = 0;\ i = 0, 1, \ldots, k), \\ R^{(j)}(t) &= P^{(j)}(\varphi(t)) \qquad (t = 1;\ j = 0, 1, \ldots, l), \end{aligned}\right\} \tag{1.20}$$

where $\varphi : [0,\, 1] \to [0,\, 1]$ is a strictly increasing function with $\varphi(0) = 0$ and $\varphi(1) = 1$.

Notice that, in general, curves which are $G^{k,l}$-continuous can be $C^{k,l}$-continuous after a proper reparametrization. Clearly, $C^{k,l}$ continuity implies $G^{k,l}$ continuity. However, the converse, is not necessarily true. Therefore, geometric continuity is more general and there is a certain flexibility which leaves some room for further optimization. Problem 1.9 has been recently discussed in several papers (see [44, 68, 70, 74, 76, 77, 91, 92, 113, 114]). The vast majority of them simplify the problem by assuming that $\varphi'(0) = \varphi'(1) = 1$, which implies, e.g., the *hybrid $C^{1,1}/G^{2,2}$-constrained degree reduction*, meaning that we impose constraints of $C^{1,1}$ continuity, followed by $G^{2,2}$ continuity, at the endpoints. As it turns out, it is possible

to apply an extended version of the method by Woźny and Lewanowicz [103] as an essential part of the algorithm of solving Problem 1.9. Such an approach is proposed in the author's joint work with Lewanowicz and Woźny [44]. It is worth mentioning that this is the only paper dealing with Problem 1.9 in its most general form. For details, see Chapter 4. More information on the $G^{k,l}$ continuity can be found in §2.2. In addition, we also discuss in detail the *hybrid continuity constraints* (also known as $C^{p,q}/G^{k,l}$ *continuity constraints*). See §2.3.

As we have seen, *conventional degree reduction of Bézier curves* is to minimize a chosen error function subject to some continuity constraints at the endpoints. As a result of this common strategy, one may obtain control points which are located far away from the plot of the curve. Consequently, it may happen that further editing of the resulting curve can be difficult or even impossible. In Chapter 5, which is based on papers [43, 46], we give appropriate examples to illustrate this issue. Then, we propose a new approach to the problem of $C^{k,l}$-constrained degree reduction of planar Bézier curves by imposing the so-called *box constraints*. Next, we present two methods of solving the new problem. They are completely different than in the case of the conventional approach. Finally, we obtain curves which are suitable for further modification and applications.

1.6.2 Merging of Bézier curves

Conventional merging of Bézier curves is to approximate a composite Bézier curve with a single Bézier curve which minimizes a selected error function and satisfies certain continuity constraints at the endpoints.

As already noted, degree reduction of Bézier curves has been extensively studied. In contrast, the number of articles dealing with the problem of merging is rather limited (see [19, 20, 45, 50, 51, 69, 71, 72, 75, 96, 102, 115]). It is noteworthy that papers [19, 96] deal with *merging of B-spline curves*. A B-spline curve is a generalization of a Bézier curve (see, e.g., [85, §3]). Nevertheless, further on in this thesis, we limit ourselves to Bézier curves.

Notice that in papers [20, 50, 51, 69, 72], the so-called l_2*-norm* is used (see, e.g., [72, (7)]), which is simpler than the L_2-norm. However, according to the comparison by Lu [72, §5], the latter is a better option. Therefore, further on in this thesis, we focus on the minimization of the L_2-error.

As it turns out, many observations concerning the degree reduction problem apply to the merging problem as well. Taking into account some of the previous remarks, we formulate the following problem of $C^{k,l}$*-constrained merging of Bézier curves*.

Problem 1.10. *[$C^{k,l}$-constrained merging of Bézier curves]*
Let $0 = t_0 < t_1 < \ldots < t_s = 1$ be a partition of the interval $[0, 1]$. Let there be given a composite Bézier curve $P(t)$ ($t \in [0, 1]$) which in the interval $[t_{i-1}, t_i]$ ($i = 1, 2, \ldots, s$) is exactly represented as a Bézier curve $P^i(t)$ of degree n_i, i.e.,

$$P(t) = P^i(t) := \sum_{j=0}^{n_i} p_j^i \, B_j^{n_i}\left(\frac{t - t_{i-1}}{\Delta t_{i-1}}\right) \qquad (t \in [t_{i-1}, t_i]). \tag{1.21}$$

Find a Bézier curve of degree m ($\geq \max_i n_i$)

$$R(t) := \sum_{j=0}^{m} r_j \, B_j^m(t) \qquad (t \in [0, 1]) \tag{1.22}$$

so that the following conditions are satisfied:

(i) L_2-error

$$\|P - R\|_{L_2} = \sqrt{\int_0^1 \|P(t) - R(t)\|^2 \, \mathrm{d}t} \tag{1.23}$$

is minimized in the space Π_m^d;

(ii) P and R are $C^{k,l}$-continuous at the endpoints, i.e.,

$$\left.\begin{aligned} R^{(i)}(0) &= P^{(i)}(0) \qquad (i = 0, 1, \ldots, k), \\ R^{(j)}(1) &= P^{(j)}(1) \qquad (j = 0, 1, \ldots, l), \end{aligned}\right\} \tag{1.24}$$

where $k \leq n_1$, $l \leq n_s$, $k, l \geq -1$ and $k + l < m - 1$.

Problems of the above type are discussed in [51, 72, 102]. Hu et al. [51] deal with merging of only two Bézier curves. Obviously, to merge more than two curves, one could use this algorithm repeatedly. Unfortunately, such an approach increases the error of the approximation as well as the computational cost. The other methods specialize in merging of more than two Bézier curves at the same time. However, the approach of Lu [72] is based on solving a system of normal equations. As in the case of degree reduction, this is a very straightforward strategy which is *computationally inefficient*, i.e., the complexity is $O(sm^3)$. Moreover, it suffers from poor numerical properties. In [102], which is the author's joint work with Woźny and Lewanowicz, a more sophisticated approach is proposed. It is based on the properties of dual Bernstein basis polynomials. Consequently, the complexity of the algorithm is $O(sm^2)$, the least among the existing algorithms. For details, see Chapter 6.

As already mentioned, the $C^{k,l}$ constraints (1.24) can be generalized, i.e., one can impose the $G^{k,l}$ constraints instead. For that reason, we formulate the problem of *$G^{k,l}$-constrained merging of Bézier curves.*

Problem 1.11. *[$G^{k,l}$-constrained merging of Bézier curves]*
Let $0 = t_0 < t_1 < \ldots < t_s = 1$ be a partition of the interval $[0, 1]$. Let there be given a composite Bézier curve $P(t)$ $(t \in [0, 1])$ which in the interval $[t_{i-1}, t_i]$ $(i = 1, 2, \ldots, s)$ is exactly represented as a Bézier curve $P^i(t)$ of degree n_i, i.e.,

$$P(t) = P^i(t) := \sum_{j=0}^{n_i} p_j^i \, B_j^{n_i}\left(\frac{t - t_{i-1}}{\Delta t_{i-1}}\right) \qquad (t \in [t_{i-1}, t_i]). \tag{1.25}$$

Find a Bézier curve of degree m $(\geq \max_i n_i)$

$$R(t) := \sum_{j=0}^{m} r_j \, B_j^m(t) \qquad (t \in [0, 1]) \tag{1.26}$$

so that the following conditions are satisfied:

(i) L_2-error

$$\|P - R\|_{L_2} = \sqrt{\int_0^1 \|P(t) - R(t)\|^2 \, \mathrm{d}t}$$

is minimized in the space Π_m^d;

(ii) P and R are $G^{k,l}$-continuous at the endpoints, i.e.,

$$\left.\begin{aligned} R^{(i)}(t) &= P^{(i)}(\varphi(t)) \qquad (t=0;\ i=0,1,\ldots,k),\\ R^{(j)}(t) &= P^{(j)}(\varphi(t)) \qquad (t=1;\ j=0,1,\ldots,l), \end{aligned}\right\} \tag{1.27}$$

where $\varphi : [0,\,1] \to [0,\,1]$ is a strictly increasing function with $\varphi(0) = 0$ and $\varphi(1) = 1$. Additionally, we assume that $-1 \le k,\, l \le 3$, $k \le n_1$, $l \le n_s$ and $k + l < m - 1$.

There are only four articles which are relevant to this problem (see [71, 72, 75, 115]). However, papers [71, 115] deal with merging of only two Bézier curves. Furthermore, Lu [71] simplifies the problem by setting $\varphi'(0) = \varphi'(1) = 1$, hence the *loss of generality*. In addition, only the *symmetric continuity cases* are considered, i.e., $k = l$. Zhu and Wang [115] limit themselves to the cases of $k = l = 1$ and $k = l = 2$. In [72], Lu solves the problem of $G^{k,l}$-constrained merging of multiple adjacent Bézier curves. However, the assumption that $k = l = 1$ is made. As a result, the method has a very limited applicability in CAGD. Lu's approach [72] is generalized in [75], where one can find a method working for $k = l = 2$. Regardless of how many curves are merged, the strategy is to solve a system of normal equations. In Chapter 7, we generalize the approach given in Chapter 6 and deal efficiently with Problem 1.11 in its most general form. The material is the author's independent work and it has not been published before.

In Chapter 8, which is based on the author's joint work with Woźny [45], we propose a novel approach to the problem of $C^{k,l}$-constrained merging of planar Bézier curves. As in the case of degree reduction, we notice that resulting control points can be located far away from the plot of the curve. This time, the defect seems to be even more significant. Once again, we impose the box constraints which appear for the first time in the context of the merging problem. As a result, the merged curve is more useful in practical applications.

1.7 Outline of the thesis

To begin with, we relate the parametric and geometric continuity conditions with the control points of curves P and R (see Chapter 2). This is the first step to solve the problems of degree reduction and merging. In Chapter 3, we introduce the general concept of dual bases and show how to construct and modify them efficiently. Those methods are useful in solving the problem of degree reduction of Bézier curves with box constrains (see Chapter 5). Moreover, we focus on some of the properties of dual Bernstein polynomials, which we use to solve the conventional problems of degree reduction and merging (see Chapters 4, 6 and 7).

In Chapter 4, we generalize the approach of Woźny and Lewanowicz [103] in order to deal efficiently with Problem 1.9 in its most general form. Additionally, the simplified version of Problem 1.9, namely the degree reduction with hybrid constraints (see §2.3), is also solved. Next, a new approach to the problem of $C^{k,l}$-constrained degree reduction of planar Bézier curves is proposed (see Chapter 5). We impose the box constraints and explain the purpose of those restrictions. The new degree reduction problem requires completely different methods than the conventional ones. We present two of them, and show that the best option is to use the so-called BVLS algorithm combined with fast methods of construction and modification of dual bases.

Chapter 6 brings a complete solution of Problem 1.10. Using fast connections between Bernstein and dual Bernstein polynomials, we obtain efficient merging method having the

lowest computational complexity among all existing methods. In Chapter 7, we apply an extended version of this method as an essential part of the algorithms of solving Problem 1.11 with and without the simplifying assumptions (see §2.3). In Chapter 8, the problem of merging of planar Bézier curves with box constraints is formulated and solved. As in the case of the box-constrained degree reduction, this new idea is justified by some illustrative examples.

Chapter 2

Continuity constraints

As it was stated in the previous chapter, a single Bézier curve is not flexible enough to fit more complex shapes. To handle this issue properly, one should use a composite Bézier curve which consists of many Bézier curves joined end-to-end. However, one must guarantee smoothness at the places where the pieces connect. A composite Bézier curve is smooth if its derivatives, up to some order, are properly defined. There are two main concepts, namely parametric and geometric continuity. Parametric continuity implies continuous derivatives. It guarantees smoothness of a curve and of its parametrization. For instance, C^1 continuity means that, in addition to C^0 continuity, both *tangent vectors* must have the same *direction* and *magnitude*. Geometric continuity is defined as agreement of derivatives after a suitable reparametrization. In other words, it involves certain relaxation of the parametrization. Nevertheless, the connection stays smooth. For instance, G^1 continuity means that, in addition to G^0 continuity, both tangent vectors point in the same direction, however, their magnitudes may be different. According to Farin [33, §11.1], "The concept of geometric continuity is more appropriate when dealing with shape; parametric continuity is appropriate when speed of traversal is an issue." Therefore, in many cases, parametric continuity seems unnecessary and too restrictive. Geometric continuity, on the other hand, might be a better option. For a detailed explanation, see the pioneering works by Barsky and DeRose [6, 27].

When dealing with a composite Bézier curve, even a minor modification of a single segment may affect the continuity. For that reason, operations such as degree reduction and merging are performed under certain continuity conditions (see Problems 1.8–1.11). In this chapter, a primary objective is to relate the continuity constraints with the control points.

2.1 Parametric continuity constraints

First, we recall the parametric continuity constraints associated with $C^{k,l}$-constrained degree reduction of Bézier curves (see Problem 1.8),

$$\left.\begin{aligned} R^{(i)}(0) &= P^{(i)}(0) \qquad (i = 0, 1, \ldots, k), \\ R^{(j)}(1) &= P^{(j)}(1) \qquad (j = 0, 1, \ldots, l). \end{aligned}\right\} \tag{2.1}$$

Now, using the formulas (1.8) and the explicit definition of the forward difference operator (see (1.7), cf. (1.6)), we get

$$\begin{aligned} R^{(i)}(0) &= \frac{m!}{(m-i)!}\Delta^i r_0 = \frac{m!}{(m-i)!}\sum_{h=0}^{i}(-1)^{i+h}\binom{i}{h}r_h, \\ R^{(j)}(1) &= \frac{m!}{(m-j)!}\Delta^j r_{m-j} = \frac{m!}{(m-j)!}\sum_{h=0}^{j}(-1)^{j+h}\binom{j}{h}r_{m-j+h}. \end{aligned} \tag{2.2}$$

Next, we substitute the formulas (1.8) and (2.2) into the equations (2.1). Finally, some simple manipulations lead to the following well-known formulas for the control points $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$ (see, e.g., [103, §4]):

$$r_j = \binom{n}{j}\binom{m}{j}^{-1}\Delta^j p_0 - \sum_{h=0}^{j-1}(-1)^{j+h}\binom{j}{h}r_h \qquad (j = 0, 1, \ldots, k), \tag{2.3}$$

$$r_{m-j} = (-1)^j\binom{n}{j}\binom{m}{j}^{-1}\Delta^j p_{n-j} - \sum_{h=1}^{j}(-1)^h\binom{j}{h}r_{m-j+h} \qquad (j = 0, 1, \ldots, l), \tag{2.4}$$

respectively.

Now, let us consider the problem of $C^{k,l}$-constrained merging of Bézier curves (see Problem 1.10). Here the $C^{k,l}$ conditions have the following interpretation: R and P^1 are C^k-continuous at $t = 0$, while R and P^s are C^l-continuous at $t = 1$. Notice that only the first and the last segments of the original composite Bézier curve are considered, hence the additional assumptions that $k \leq n_1$ and $l \leq n_s$. Proceeding analogously as before, we obtain

$$r_j = t_1^{-j}\binom{n_1}{j}\binom{m}{j}^{-1}\Delta^j p_0^1 - \sum_{h=0}^{j-1}(-1)^{j+h}\binom{j}{h}r_h \qquad (j = 0, 1, \ldots, k), \tag{2.5}$$

$$r_{m-j} = (t_{s-1}-1)^{-j}\binom{n_s}{j}\binom{m}{j}^{-1}\Delta^j p_{n_s-j}^s - \sum_{h=1}^{j}(-1)^h\binom{j}{h}r_{m-j+h} \qquad (j = 0, 1, \ldots, l) \tag{2.6}$$

(cf. (2.3) and (2.4)).

Remark 2.1. While deriving the formulas (2.5) and (2.6), the authors of [45, 102] forgot that the parametrization of P^1 and P^s is $[0, t_1]$ and $[t_{s-1}, 1]$, respectively. Instead, they used $[0, 1]$ for both curves. Consequently, the formulas given there are slightly different than (2.5), (2.6); and resulting curves must be slightly different as well. Note that during the derivation of (2.5) and (2.6), some researchers choose the interval $[0, 1]$ for P^1 and P^s (see, e.g., [71, Remark 2.3]).

2.2 Geometric continuity constraints

In this subsection, we relate the geometric continuity constraints with the control points. Limiting ourselves to the cases of $-1 \leq k, l \leq 3$, which are useful from a practical point of view, we discuss the $G^{k,l}$ conditions in detail.

First, we focus on the problem of $G^{k,l}$-constrained degree reduction of Bézier curves (see Problem 1.9). Recall the geometric continuity constraints,

$$\left.\begin{aligned} R^{(i)}(t) &= P^{(i)}(\varphi(t)) \qquad (t=0;\ i=0,1,\ldots,k),\\ R^{(j)}(t) &= P^{(j)}(\varphi(t)) \qquad (t=1;\ j=0,1,\ldots,l). \end{aligned}\right\} \tag{2.7}$$

It can be easily shown that

$$\frac{\mathrm{d}^i}{\mathrm{d}t^i}P(\varphi(t)) = \sum_{j=1}^{i} U_{ij}\frac{\mathrm{d}^j}{\mathrm{d}u^j}P(u)\Big|_{u=\varphi(t)}, \tag{2.8}$$

where the coefficients $U_{ij} \equiv U_{ij}(t)$ depend only on the derivatives $\delta_j \equiv \delta_j(t) := \varphi^{(j)}(t)$ for $j = 1, 2, \ldots, i$. For instance,

$$\begin{aligned} &U_{11} = \delta_1;\\ &U_{21} = \delta_2, \quad U_{22} = \delta_1^2;\\ &U_{31} = \delta_3, \quad U_{32} = 3\delta_1\delta_2, \qquad\quad U_{33} = \delta_1^3;\\ &U_{41} = \delta_4, \quad U_{42} = 3\delta_2^2 + 4\delta_1\delta_3, \quad U_{43} = 6\delta_1^2\delta_2, \quad U_{44} = \delta_1^4. \end{aligned}$$

Now, applying the formulas (1.8), (2.2) and (2.8) to the equations (2.7), we obtain the following recursive forms for the control points $r_0, r_1, \ldots, r_k$:

$$\begin{aligned} &r_0 = p_0,\\ &r_i = \frac{(-1)^i}{(-m)_i}\sum_{j=1}^{i}(-1)^j(-n)_j U_{ij}(0)\Delta^j p_0 - \sum_{h=0}^{i-1}(-1)^{i+h}\binom{i}{h}r_h \qquad (i=1,2,\ldots,k), \end{aligned} \tag{2.9}$$

and similarly for $r_m, r_{m-1}, \ldots, r_{m-l}$:

$$\begin{aligned} &r_m = p_n,\\ &r_{m-i} = \frac{1}{(-m)_i}\sum_{j=1}^{i}(-1)^j(-n)_j U_{ij}(1)\Delta^j p_{n-j} - \sum_{h=1}^{i}(-1)^h\binom{i}{h}r_{m-i+h} \qquad (i=1,2,\ldots,l), \end{aligned} \tag{2.10}$$

where *shifted factorial* is defined by

$$(c)_0 := 1, \quad (c)_h := c(c+1)\ldots(c+h-1) \quad (h \geq 1;\ c \in \mathbb{C}).$$

Observe that the control points r_1, $r_2, \ldots,$ r_k depend on the parameters

$$\lambda_i := \delta_i(0) = \varphi^{(i)}(0) \qquad (i=1,2,\ldots,k),$$

while the points $r_{m-1}, r_{m-2}, \ldots, r_{m-l}$ depend on

$$\mu_j := \delta_j(1) = \varphi^{(j)}(1) \qquad (j=1,2,\ldots,l).$$

Remark 2.2. Note that $G^{k,l} \equiv C^{k,l}$ for $-1 \leq k,\, l < 1$.

Remark 2.3. In contrast to the parametric case, the control points (2.9) and (2.10) are not yet known since the parameters $\{\lambda_i\}$ and $\{\mu_j\}$ are not given in advance. Note that the $C^{k,l}$ constraints imply that $\lambda_1 = \mu_1 := 1$ and $\lambda_i = \mu_j := 0$ $(i, j > 1)$.

For $k, l > 3$, the constraints (2.7) are overly restrictive. Therefore, in practice, it is sufficient to consider the cases of $-1 \leq k, l \leq 3$. When $k = 3$, we have:

$$r_0 = p_0, \quad r_1 = p_0 + \frac{n}{m}\lambda_1 \Delta p_0, \tag{2.11}$$

$$r_2 = p_0 + \frac{n}{m}\left[2\lambda_1 + \frac{1}{m-1}\lambda_2\right]\Delta p_0 + \frac{(n-1)_2}{(m-1)_2}\lambda_1^2 \Delta^2 p_0, \tag{2.12}$$

$$\begin{aligned} r_3 = p_0 &+ \frac{n}{m}\left[3\lambda_1 + \frac{3}{m-1}\lambda_2 + \frac{1}{(m-2)_2}\lambda_3\right]\Delta p_0 \\ &+ 3\frac{(n-1)_2}{(m-1)_2}\left[\lambda_1^2 + \frac{1}{m-2}\lambda_1\lambda_2\right]\Delta^2 p_0 + \frac{(n-2)_3}{(m-2)_3}\lambda_1^3 \Delta^3 p_0. \end{aligned} \tag{2.13}$$

In the case of $k = 2$, we use (2.11) and (2.12). For $k = 1$, the formulas (2.11) hold. Analogously, when $l = 3$, we have:

$$r_m = p_n, \quad r_{m-1} = p_n - \frac{n}{m}\mu_1 \Delta p_{n-1}, \tag{2.14}$$

$$r_{m-2} = p_n - \frac{n}{m}\left[2\mu_1 - \frac{1}{m-1}\mu_2\right]\Delta p_{n-1} + \frac{(n-1)_2}{(m-1)_2}\mu_1^2 \Delta^2 p_{n-2}, \tag{2.15}$$

$$\begin{aligned} r_{m-3} = p_n &- \frac{n}{m}\left[3\mu_1 - \frac{3}{m-1}\mu_2 + \frac{1}{(m-2)_2}\mu_3\right]\Delta p_{n-1} \\ &+ 3\frac{(n-1)_2}{(m-1)_2}\left[\mu_1^2 - \frac{1}{m-2}\mu_1\mu_2\right]\Delta^2 p_{n-2} - \frac{(n-2)_3}{(m-2)_3}\mu_1^3 \Delta^3 p_{n-3}. \end{aligned} \tag{2.16}$$

In the case of $l = 2$, we use (2.14) and (2.15). For $l = 1$, the formulas (2.14) hold.

Now, let us look at the problem of $G^{k,l}$-constrained merging of Bézier curves (see Problem 1.11). Here the $G^{k,l}$ conditions have the following interpretation: R and P^1 are G^k-continuous at $t = 0$, while R and P^s are G^l-continuous at $t = 1$. Once again, observe that only the first and the last segments of the original composite Bézier curve are considered. Similarly as before, we limit ourselves to the cases of $-1 \leq k, l \leq 3$. When $k = 3$, we have:

$$r_0 = p_0^1, \quad r_1 = p_0^1 + \frac{n_1}{m}t_1^{-1}\lambda_1 \Delta p_0^1, \tag{2.17}$$

$$r_2 = p_0^1 + \frac{n_1}{m}t_1^{-1}\left[2\lambda_1 + \frac{1}{m-1}\lambda_2\right]\Delta p_0^1 + \frac{(n_1-1)_2}{(m-1)_2}t_1^{-2}\lambda_1^2 \Delta^2 p_0^1, \tag{2.18}$$

$$\begin{aligned} r_3 = p_0^1 &+ \frac{n_1}{m}t_1^{-1}\left[3\lambda_1 + \frac{3}{m-1}\lambda_2 + \frac{1}{(m-2)_2}\lambda_3\right]\Delta p_0^1 \\ &+ 3\frac{(n_1-1)_2}{(m-1)_2}t_1^{-2}\left[\lambda_1^2 + \frac{1}{m-2}\lambda_1\lambda_2\right]\Delta^2 p_0^1 + \frac{(n_1-2)_3}{(m-2)_3}t_1^{-3}\lambda_1^3 \Delta^3 p_0^1, \end{aligned} \tag{2.19}$$

(cf. (2.11)–(2.13)). In the case of $k = 2$, we use (2.17) and (2.18). For $k = 1$, the formulas (2.17) hold. Analogously, when $l = 3$, we have:

$$r_m = p^s_{n_s}, \quad r_{m-1} = p^s_{n_s} - \frac{n_s}{m}(1 - t_{s-1})^{-1}\mu_1 \Delta p^s_{n_s-1}, \tag{2.20}$$

$$r_{m-2} = p^s_{n_s} - \frac{n_s}{m}(1 - t_{s-1})^{-1}\left[2\mu_1 - \frac{1}{m-1}\mu_2\right]\Delta p^s_{n_s-1} + \frac{(n_s-1)_2}{(m-1)_2}(1 - t_{s-1})^{-2}\mu_1^2\Delta^2 p^s_{n_s-2}, \tag{2.21}$$

$$\begin{aligned} r_{m-3} = p^s_{n_s} &- \frac{n_s}{m}(1 - t_{s-1})^{-1}\left[3\mu_1 - \frac{3}{m-1}\mu_2 + \frac{1}{(m-2)_2}\mu_3\right]\Delta p^s_{n_s-1} \\ &+ 3\frac{(n_s-1)_2}{(m-1)_2}(1 - t_{s-1})^{-2}\left[\mu_1^2 - \frac{1}{m-2}\mu_1\mu_2\right]\Delta^2 p^s_{n_s-2} \\ &- \frac{(n_s-2)_3}{(m-2)_3}(1 - t_{s-1})^{-3}\mu_1^3\Delta^3 p^s_{n_s-3}, \end{aligned} \tag{2.22}$$

(cf. (2.14)–(2.16)). In the case of $l = 2$, we use (2.20) and (2.21). For $l = 1$, the formulas (2.20) hold.

As in the case of $G^{k,l}$-constrained degree reduction, optimal values for the parameters $\{\lambda_i\}$ and $\{\mu_j\}$ are not known in advance.

2.3 Hybrid continuity constraints

Notice that the control points (2.12), (2.13), (2.15), (2.16), (2.18), (2.19), (2.21) and (2.22) are *nonlinear polynomial functions* of the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$. As we shall see in Chapters 4 and 7, this is a serious issue which makes further computations more difficult and expensive. However, if we set $\lambda_1 = \mu_1 := 1$, which implies $C^{1,1}$ continuity (see Remark 2.3), then the previously mentioned functions become *linear*. This idea is widely used in degree reduction (see, e.g., [92]) and merging (see, e.g., [71]).

In general, we use $C^{p,q}/G^{k,l}$ notation to describe the so-called hybrid constraints, where p, $q \in \{-, 1\}$ and ($k \geq 2$ or $l \geq 2$). In the case of $k \geq 2$ and $p = 1$, we set $\lambda_1 := 1$. Similarly, for $l \geq 2$ and $q = 1$, we set $\mu_1 := 1$. Setting $p = q := -$ means that we do not fix λ_1, μ_1, respectively. Clearly, $C^{-,-}/G^{k,l}$ denotes $G^{k,l}$.

We denote the above-described approach to the problems of degree reduction and merging of Bézier curves as *$C^{p,q}/G^{k,l}$-constrained degree reduction of Bézier curves* and *$C^{p,q}/G^{k,l}$-constrained merging of Bézier curves*, respectively. For further details, see Chapters 4 and 7.

Chapter 3

Dual bases

Most of the algorithms given in this thesis are based on the properties of *dual polynomial bases*. Therefore, in this chapter, we introduce the concept of *dual bases* (see §3.1). In §3.2, we give some methods of construction of dual bases in general. Finally, we discuss in detail the properties of dual Bernstein polynomials (see §3.3).

3.1 Introduction

Let $B_n := \{b_0, b_1, \ldots, b_n\}$ be a basis of the linear space $\mathcal{B}_n := \operatorname{span} B_n$. A *dual basis* $D_n := \left\{d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)}\right\}$ for the basis B_n of the space $\mathcal{B}_n$ satisfies the following *duality conditions*:

$$\operatorname{span} D_n = \mathcal{B}_n, \tag{3.1}$$

$$\left\langle b_i, d_j^{(n)} \right\rangle = \delta_{ij} \qquad (i, j = 0, 1, \ldots, n), \tag{3.2}$$

where $\langle \cdot, \cdot \rangle : \mathcal{B}_n \times \mathcal{B}_n \to \mathbb{C}$ is an *inner product*, and $d_j^{(n)}$ $(j = 0, 1, \ldots, n)$ are called *dual functions*.

Now, we present some well-known facts about dual bases.

Fact 3.1 ([100])**.** *Every $f_n \in \mathcal{B}_n$ can be written in the following way:*

$$f_n = \sum_{i=0}^{n} \left\langle f_n, d_i^{(n)} \right\rangle b_i.$$

Proof. First, we write

$$f_n = \sum_{i=0}^{n} a_i b_i \qquad (a_i \in \mathbb{C})$$

since every $f_n \in \mathcal{B}_n$ has a unique representation in the basis B_n. Now, let us consider

$$\left\langle f_n, d_j^{(n)} \right\rangle = \sum_{i=0}^{n} a_i \left\langle b_i, d_j^{(n)} \right\rangle \qquad (j = 0, 1, \ldots, n).$$

Obviously, the duality conditions (3.2) imply that

$$\left\langle f_n, d_j^{(n)} \right\rangle = a_j \qquad (j = 0, 1, \ldots, n).$$

□

Fact 3.2 ([100]). *Given a function g,*

$$f_n^* = \sum_{i=0}^{n} \left\langle g, d_i^{(n)} \right\rangle b_i \tag{3.3}$$

is the best least squares approximation of g in the space $\mathcal{B}_n$, i.e.,

$$\|g - f_n^*\|_2 = \min_{f_n \in \mathcal{B}_n} \|g - f_n\|_2,$$

where $\|\cdot\|_2 := \sqrt{\langle \cdot, \cdot \rangle}$ is the least squares norm.

Proof. According to Fact 3.1, f_n^* has the following representation in the basis B_n of the space $\mathcal{B}_n$:

$$f_n^* = \sum_{i=0}^{n} \left\langle f_n^*, d_i^{(n)} \right\rangle b_i.$$

On the other hand, a classical characterization of the best least squares approximation is that $\langle g - f_n^*, h \rangle = 0$ holds for any $h \in \mathcal{B}_n$. In particular, for $h = d_i^{(n)}$ $(i = 0, 1, \ldots, n)$, we obtain

$$\left\langle g, d_i^{(n)} \right\rangle = \left\langle f_n^*, d_i^{(n)} \right\rangle \qquad (i = 0, 1, \ldots, n).$$

Hence, the formula (3.3) follows. □

Recently, dual bases with their applications in numerical analysis and CAGD have been extensively studied. Dual Bernstein polynomials (see, e.g., [21, 52, 62, 63, 87, 88]) have found their application in the algorithm of computing roots of polynomials (see [7]), degree reduction of Bézier curves (see [44, 73, 103]), merging of Bézier curves (see [102]) and polynomial approximation of rational Bézier curves (see [65, 66]). *Bivariate dual Bernstein polynomials* can be used in degree reduction of triangular Bézier surfaces (see [104]) and approximation of rational triangular Bézier surfaces by polynomial triangular Bézier surfaces (see [61]). In order to perform degree reduction of tensor product Bézier surfaces, one can use the properties of *dual tensor product Bernstein polynomials* (see [64]). In [101], one can find some results on *dual B-spline functions*. Paper [48] deals with the construction of *dual B-spline functionals*. *Dual Wang-Bézier* and *dual Bézier-Said-Wang type generalized Ball polynomials* have also attracted a lot of attention lately (see [4, 106–108]). The theory and applications of *dual NS-power bases* are given in [109]. Dual polynomial bases were studied in [41]. Dual basis functions in subspaces of inner product spaces were discussed in [53]. As for the properties of dual bases in general, see [46, 100, 101].

3.2 Construction of dual bases

In this subsection, we show how to construct a dual basis.

3.2.1 A straightforward method

To begin with, let us focus on the most obvious approach (see, e.g., [100, §2]). A goal is to represent each dual function $d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)}$ in the basis B_n, i.e., we write

$$d_i^{(n)} = \sum_{j=0}^{n} a_{ij}^{(n)} b_j \qquad (i = 0, 1, \ldots, n)$$

and look for the coefficients $a_{ij}^{(n)}$ $(i,j=0,1,\ldots,n)$. Such a representation exists since $d_i^{(n)} \in \mathcal{B}_n$ (cf. (3.1)). Now, for each $i \in \{0,1,\ldots,n\}$, the duality conditions (3.2) yield the following system of linear equations:

$$\delta_{ik} = \left\langle d_i^{(n)}, b_k \right\rangle = \sum_{j=0}^{n} a_{ij}^{(n)} \langle b_j, b_k \rangle \qquad (k=0,1,\ldots,n).$$

Finally, we solve each system for the coefficients $a_{ij}^{(n)}$ $(j=0,1,\ldots,n)$. Note that these systems share the same *Gramian matrix* which must be inverted.

3.2.2 An orthonormal basis approach

Let $q_0, q_1, \ldots, q_n$ be an *orthonormal basis* of the space $\mathcal{B}_n$ with respect to the inner product $\langle \cdot, \cdot \rangle$, i.e., the following conditions are satisfied:

$$\operatorname{span}\{q_0, q_1, \ldots, q_n\} = \mathcal{B}_n,$$

$$\langle q_i, q_j \rangle = \delta_{ij} \qquad (i,j=0,1,\ldots,n).$$

As it turns out, orthonormal and dual bases are related.

Theorem 3.3 ([62])**.** *Suppose that we know a representation of each $q_0, q_1, \ldots, q_n$ in the basis $b_0, b_1, \ldots, b_n$,*

$$q_i = \sum_{j=0}^{n} h_{ij} b_j \qquad (i=0,1,\ldots,n). \tag{3.4}$$

Then, the dual functions $d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)}$ can be written in the following way:

$$d_j^{(n)} = \sum_{i=0}^{n} h_{ij} q_i \qquad (j=0,1,\ldots,n).$$

Unfortunately, the representation (3.4), as well as the orthonormal basis itself, is in many cases unknown.

3.2.3 A more sophisticated method ($D_n \Longrightarrow D_{n+1}$)

Now, we describe a more sophisticated method of construction of dual bases.

Suppose that B_n is the given basis of the space $\mathcal{B}_n$ and the dual basis D_n with respect to the inner product $\langle \cdot, \cdot \rangle$ is known as well. In [101], Woźny proposed an efficient method of constructing the dual basis

$$D_{n+1} := \left\{ d_0^{(n+1)}, d_1^{(n+1)}, \ldots, d_{n+1}^{(n+1)} \right\}$$

for $B_{n+1} := B_n \cup \{b_{n+1}\}$. See also his previous method [100]. Note that the results which were first published in [101] contain some mistakes because the author forgot to use a complex conjugate of certain coefficients (see [46, Remark 2.3]). However, in the case of a *real-valued inner product*, the formulas given there are true. The corrected results were published in [46]. Further on in this subsection, we present the connection between D_n and D_{n+1} as well as the algorithm of constructing D_{n+1}.

Theorem 3.4 ([101])**.** *The dual functions from D_n and D_{n+1} are related in the following way:*

$$d_i^{(n+1)} = d_i^{(n)} - w_i^{(n+1)} d_{n+1}^{(n+1)} \qquad (i = 0, 1, \ldots, n), \tag{3.5}$$

where

$$w_i^{(n+1)} := \left\langle d_i^{(n)}, b_{n+1} \right\rangle. \tag{3.6}$$

As a result of Theorem 3.4, each dual function $d_i^{(n+1)}$ $(i = 0, 1, \ldots, n)$ depends on $d_i^{(n)}$, which is known, and on $d_{n+1}^{(n+1)}$ which must be computed. Note that $\operatorname{span} B_{n+1} = \operatorname{span}(D_n \cup \{b_{n+1}\})$. Therefore, we can write

$$d_{n+1}^{(n+1)} = \sum_{h=0}^{n} c_h^{(n+1)} d_h^{(n)} + c_{n+1}^{(n+1)} b_{n+1}, \tag{3.7}$$

and solve the following system of linear equations:

$$\begin{cases} 0 = \left\langle d_{n+1}^{(n+1)}, b_i \right\rangle = c_i^{(n+1)} + c_{n+1}^{(n+1)} v_i^{(n+1)} \qquad (i = 0, 1, \ldots, n), \\ 1 = \left\langle d_{n+1}^{(n+1)}, b_{n+1} \right\rangle = \displaystyle\sum_{h=0}^{n} c_h^{(n+1)} w_h^{(n+1)} + c_{n+1}^{(n+1)} v_{n+1}^{(n+1)}, \end{cases}$$

where

$$v_j^{(n+1)} := \langle b_{n+1}, b_j \rangle \qquad (j = 0, 1, \ldots, n+1), \tag{3.8}$$

for the coefficients $c_0^{(n+1)}, c_1^{(n+1)}, \ldots, c_{n+1}^{(n+1)}$. According to [101, §2], the solution is simple, namely

$$c_{n+1}^{(n+1)} = \left(v_{n+1}^{(n+1)} - \sum_{h=0}^{n} v_h^{(n+1)} w_h^{(n+1)} \right)^{-1}, \tag{3.9}$$

$$c_h^{(n+1)} = -v_h^{(n+1)} c_{n+1}^{(n+1)} \qquad (h = 0, 1, \ldots, n). \tag{3.10}$$

Corollary 3.5 ([101])**.** Let $f_n^* \in \mathcal{B}_n$ be the best least squares approximation of a function g in the space $\mathcal{B}_n$, i.e.,

$$\|g - f_n^*\|_2 = \min_{f_n \in \mathcal{B}_n} \|g - f_n\|_2, \tag{3.11}$$

where

$$f_n^* = \sum_{i=0}^{n} e_i^{(n)} b_i \tag{3.12}$$

with $e_i^{(n)} := \left\langle g, d_i^{(n)} \right\rangle$ for $i = 0, 1, \ldots, n$ (see Fact 3.2). Suppose that we know the coefficients $e_i^{(n)}$ $(i = 0, 1, \ldots, n)$ and our goal is to compute the optimal element $f_{n+1}^* \in \mathcal{B}_{n+1}$ for the same function g. Observe that Fact 3.2, along with the formulas (3.5) and (3.7), yields the following relations:

$$\begin{aligned} e_{n+1}^{(n+1)} &= \sum_{h=0}^{n} \overline{c_h^{(n+1)}} e_h^{(n)} + \overline{c_{n+1}^{(n+1)}} \langle g, b_{n+1} \rangle, \\ e_i^{(n+1)} &= e_i^{(n)} - \overline{w_i^{(n+1)}} e_{n+1}^{(n+1)} \qquad (i = 0, 1, \ldots, n), \end{aligned}$$

where $\overline{z}$ is the complex conjugate of $z \in \mathbb{C}$.

The above-described idea is summarized in the following algorithm.

Algorithm 3.6 ([101]). *[$D_n \Longrightarrow D_{n+1}$]*

Input: $D_n = \left\{d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)}\right\}$, $B_{n+1} = \{b_0, b_1, \ldots, b_{n+1}\}$

Output: $D_{n+1} = \left\{d_0^{(n+1)}, d_1^{(n+1)}, \ldots, d_{n+1}^{(n+1)}\right\}$

`Step 1.` *Compute* $w_i^{(n+1)}$ $(i = 0, 1, \ldots, n)$ *by* (3.6).

`Step 2.` *Compute* $v_j^{(n+1)}$ $(j = 0, 1, \ldots, n+1)$ *by* (3.8).

`Step 3.` *Compute* $c_{n+1}^{(n+1)}$ *by* (3.9).

`Step 4.` *Compute* $c_h^{(n+1)}$ $(h = 0, 1, \ldots, n)$ *by* (3.10).

`Step 5.` *Compute* $d_{n+1}^{(n+1)}$ *by* (3.7).

`Step 6.` *Compute* $d_i^{(n+1)}$ $(i = 0, 1, \ldots, n)$ *by* (3.5).

`Step 7.` *Return the dual basis* $\left\{d_0^{(n+1)}, d_1^{(n+1)}, \ldots, d_{n+1}^{(n+1)}\right\}$.

The next algorithm computes a sequence of dual bases $D_0, D_1, \ldots, D_N$.

Algorithm 3.7 ([101]). *[Construction of dual bases $D_0, D_1, \ldots, D_N$]*

Input: $B_n = \{b_0, b_1, \ldots, b_n\}$ $(n = 0, 1, \ldots, N)$

Output: $D_n = \left\{d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)}\right\}$ $(n = 0, 1, \ldots, N)$

`Step 1.` *Set* $D_0 := \left\{\langle b_0, b_0\rangle^{-1} b_0\right\}$.

`Step 2.` *Compute* D_n $(n = 1, 2, \ldots, N)$ *using Algorithm 3.6.*

`Step 3.` *Return the dual bases* $D_0, D_1, \ldots, D_N$.

3.2.4 A new method ($D_{n+1} \Longrightarrow D_n$)

In this subsection, we prove that for the given dual basis D_{n+1}, it is possible to compute efficiently the dual basis D_n. Such a method is a result of the author's joint work with Woźny (see [46]).

Let there be given the dual basis D_{n+1}. Clearly, to compute the dual basis D_n one can try to rewrite the formula (3.5). However, note that each coefficient $w_i^{(n+1)}$ depends on the searched dual function $d_i^{(n)}$ (see (3.6)). To avoid this issue, we must find a different formula for the coefficients $w_i^{(n+1)}$ $(i = 0, 1, \ldots, n)$. In order to prove the main result, we will need the following lemma.

Lemma 3.8. *The following identity holds:*

$$\left\langle d_i^{(n)}, d_{n+1}^{(n+1)} \right\rangle = 0 \qquad (i = 0, 1, \ldots, n). \tag{3.13}$$

Proof. We use Fact 3.1 to represent each dual function $d_i^{(n)}$ as a linear combination of the elements $b_0, b_1, \ldots, b_n$,

$$d_i^{(n)} = \sum_{j=0}^{n} \left\langle d_i^{(n)}, d_j^{(n)} \right\rangle b_j.$$

Consequently, we have

$$\begin{aligned} \left\langle d_i^{(n)}, d_{n+1}^{(n+1)} \right\rangle &= \left\langle \sum_{j=0}^{n} \left\langle d_i^{(n)}, d_j^{(n)} \right\rangle b_j, d_{n+1}^{(n+1)} \right\rangle \\ &= \sum_{j=0}^{n} \left\langle d_i^{(n)}, d_j^{(n)} \right\rangle \left\langle b_j, d_{n+1}^{(n+1)} \right\rangle = 0 \end{aligned}$$

since $\left\langle b_j, d_{n+1}^{(n+1)} \right\rangle = 0$ for $j = 0, 1, \ldots, n$. □

Theorem 3.9. *The connection between the dual functions from D_n and D_{n+1} is as follows:*

$$d_i^{(n)} = d_i^{(n+1)} + w_i^{(n+1)} d_{n+1}^{(n+1)} \qquad (i = 0, 1, \ldots, n), \tag{3.14}$$

where

$$w_i^{(n+1)} := -\frac{\left\langle d_i^{(n+1)}, d_{n+1}^{(n+1)} \right\rangle}{\left\langle d_{n+1}^{(n+1)}, d_{n+1}^{(n+1)} \right\rangle} \tag{3.15}$$

(cf. Theorem 3.4).

Proof. Obviously, the relation (3.14) follows from (3.5). Now, we substitute (3.14) into the equation (3.13) and obtain

$$\begin{aligned} 0 &= \left\langle d_i^{(n+1)} + w_i^{(n+1)} d_{n+1}^{(n+1)}, d_{n+1}^{(n+1)} \right\rangle \\ &= \left\langle d_i^{(n+1)}, d_{n+1}^{(n+1)} \right\rangle + w_i^{(n+1)} \left\langle d_{n+1}^{(n+1)}, d_{n+1}^{(n+1)} \right\rangle. \end{aligned}$$

Hence, the formula (3.15) follows. □

Corollary 3.10. Let $f_{n+1}^* \in \mathcal{B}_{n+1}$ be the best least squares approximation of a function g in the space $\mathcal{B}_{n+1}$, i.e.,

$$\|g - f_{n+1}^*\|_2 = \min_{f_{n+1} \in \mathcal{B}_{n+1}} \|g - f_{n+1}\|_2,$$

where

$$f_{n+1}^* = \sum_{j=0}^{n+1} e_j^{(n+1)} b_j$$

with $e_j^{(n+1)} := \left\langle g, d_j^{(n+1)} \right\rangle$ for $j = 0, 1, \ldots, n+1$ (see Fact 3.2). Suppose that we know the coefficients $e_j^{(n+1)}$ $(j = 0, 1, \ldots, n+1)$ and our goal is to compute the optimal element $f_n^* \in \mathcal{B}_n$ for the same function g (see (3.11), (3.12)). Observe that Fact 3.2, along with the formula (3.14), yields the following relation:

$$e_i^{(n)} = e_i^{(n+1)} + \overline{w_i^{(n+1)}} e_{n+1}^{(n+1)} \qquad (i = 0, 1, \ldots, n)$$

(cf. Corollary 3.5).

Note that in contrast to (3.6), the formula (3.15) is independent of $d_i^{(n)}$ $(i = 0, 1, \ldots, n)$. Therefore, it can be used to compute the dual basis D_n, under the assumption that the dual basis D_{n+1} is given. See the following algorithm.

Algorithm 3.11. *[$D_{n+1} \Longrightarrow D_n$]*

Input: $D_{n+1} = \left\{ d_0^{(n+1)}, d_1^{(n+1)}, \ldots, d_{n+1}^{(n+1)} \right\}$

Output: $D_n = \left\{ d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)} \right\}$

`Step 1.` *For $i = 0, 1, \ldots, n$,*

(i) *compute $w_i^{(n+1)}$ by (3.15);*

(ii) *compute $d_i^{(n)}$ by (3.14).*

`Step 2.` *Return the dual basis $\left\{ d_0^{(n)}, d_1^{(n)}, \ldots, d_n^{(n)} \right\}$.*

Remark 3.12. Suppose that a dual basis of a certain space is well-known or was computed earlier. In numerical analysis and CAGD, we often look for an optimal element (in the least squares sense) which is *constrained*, e.g., by some continuity conditions. As a result, we need a dual basis of a specific subspace of the well-known space. Algorithm 3.11 can be particularly useful in those situations. For example, see Chapter 5.

As we shall see in Chapter 5, the presented algorithms are useful in CAGD.

3.3 Dual Bernstein polynomials

In this subsection, we focus on dual Bernstein polynomials, which are crucial for most of the algorithms presented in this thesis.

First, we define the inner product $\langle \cdot, \cdot \rangle_{\alpha\beta}$ by

$$\langle f, g \rangle_{\alpha\beta} := \int_0^1 (1-t)^\alpha t^\beta f(t) g(t) \mathrm{d}t \qquad (\alpha,\, \beta > -1). \tag{3.16}$$

Further on in the thesis, we assume that $\langle \cdot, \cdot \rangle_L \equiv \langle \cdot, \cdot \rangle_{00}$.

According to [52, §2], there is a unique *dual Bernstein polynomial basis of degree n*

$$D_0^n(t; \alpha, \beta), D_1^n(t; \alpha, \beta), \ldots, D_n^n(t; \alpha, \beta) \in \Pi_n,$$

associated with the Bernstein basis (1.1), so that

$$\left\langle D_i^n(\cdot; \alpha, \beta), B_j^n \right\rangle_{\alpha\beta} = \delta_{ij} \qquad (i, j = 0, 1, \ldots, n).$$

For the sake of simplicity, we set $D_i^n(t) := D_i^n(t; 0, 0)$.

Now, we consider the following restriction of the Π_n space (cf. §1.3, pt. 7). Given the integers k, l such that $k,\, l \geq -1$ and $k + l < n - 1$, let $\Pi_n^{(k,l)}$ be the space of all polynomials of degree at most n, whose derivatives of orders $0, 1, \ldots, k$ at $t = 0$, as well as derivatives of orders $0, 1, \ldots, l$ at $t = 1$, vanish,

$$\Pi_n^{(k,l)} := \left\{ P \in \Pi_n \,:\, P^{(i)}(0) = 0 \quad (0 \leq i \leq k) \text{ and } P^{(j)}(1) = 0 \quad (0 \leq j \leq l) \right\}. \tag{3.17}$$

Here we use the convention that derivative of order 0 of a function is the function itself. Clearly, $\dim \Pi_n^{(k,l)} = n-k-l-1$, and the Bernstein polynomials $B_{k+1}^n, B_{k+2}^n, \ldots, B_{n-l-1}^n$ form a basis of this space. According to [52, §3], there is a unique *constrained dual Bernstein polynomial basis of degree n*

$$D_{k+1}^{(n,k,l)}(t;\alpha,\beta), D_{k+2}^{(n,k,l)}(t;\alpha,\beta), \ldots, D_{n-l-1}^{(n,k,l)}(t;\alpha,\beta) \in \Pi_n^{(k,l)}$$

satisfying the relation $\left\langle D_i^{(n,k,l)}(\cdot;\alpha,\beta), B_j^n \right\rangle_{\alpha\beta} = \delta_{ij}$ $(i,j = k{+}1, k{+}2, \ldots, n{-}l{-}1)$. Obviously, we have $D_i^{(n,-1,-1)}(t;\alpha,\beta) = D_i^n(t;\alpha,\beta)$, which corresponds to the unconstrained case. For simplicity, we set $D_i^{(n,k,l)}(t) := D_i^{(n,k,l)}(t;0,0)$.

Further on in this subsection, we give some useful properties of the constrained and unconstrained dual Bernstein bases.

3.3.1 Connections between Bernstein and dual Bernstein bases

Now, we present some useful connections between Bernstein and dual Bernstein bases. In the context of Problems 1.10 and 1.11, the following lemmas are of great importance (see Chapters 6 and 7).

Lemma 3.13. *Let n and m be positive integers such that $n \le m$. The following formula holds:*

$$B_i^n(t) = \sum_{j=0}^{m} a_{ij}^{(n,m)} D_j^m(t) \qquad (0 \le i \le n;\ n \le m),$$

where

$$a_{ij}^{(n,m)} := \frac{1}{m+n+1} \binom{n}{i} \binom{m}{j} \binom{n+m}{i+j}^{-1}. \tag{3.18}$$

Proof. Since $\Pi_n \subset \Pi_m$, we make use of Fact 3.1 and obtain

$$a_{ij}^{(n,m)} = \langle B_i^n, B_j^m \rangle_L = \int_0^1 B_i^n(t) B_j^m(t)\,\mathrm{d}t.$$

The result follows by the well-known properties of Bernstein polynomials (see §1.3, pts. 5 and 12). □

Lemma 3.14 ([63])**.** *The constrained dual Bernstein polynomials have the following representation in the Bernstein basis:*

$$D_i^{(m,k,l)}(t) = \sum_{j=k+1}^{m-l-1} c_{ij}(m,k,l)\, B_j^m(t),$$

where the coefficients $c_{ij} \equiv c_{ij}(m,k,l)$ satisfy the recurrence relation

$$c_{i+1,j} = \frac{1}{U(i)} \left\{ 2(i-j)(i+j-m)\, c_{ij} + W(j)\, c_{i,j-1} + U(j)\, c_{i,j+1} - W(i)\, c_{i-1,j} \right\}$$

$$(k+1 \le i \le m-l-2, \quad k+1 \le j \le m-l-1) \tag{3.19}$$

with

$$U(u) := (u-m)(u-k)(u+k+2)/(u+1),$$

$$W(u) := u(u-m-l-2)(u-m+l)/(u-m-1).$$

We adopt the convention that $c_{ij} := 0$ *if* $i \le k$, *or* $i \ge m-l$, *or* $j \le k$, *or* $j \ge m-l$. *The starting values are*

$$c_{k+1,j} = (-1)^{j-k-1}(2k+3)\binom{m}{k+1}^{-1}\binom{m+k-l+1}{2k+3}\binom{m}{j}^{-1} \times \binom{m-k-l-2}{j-k-1}\binom{m+k+l+3}{k+j+2},$$

where $j = k+1, k+2, \ldots, m-l-1$.

According to Lemma 3.14, the coefficients c_{ij} can be put in the following table:

	0	0	...	0	
0	$c_{k+1,k+1}$	$c_{k+1,k+2}$	...	$c_{k+1,m-l-1}$	0
0	$c_{k+2,k+1}$	$c_{k+2,k+2}$	...	$c_{k+2,m-l-1}$	0
...					
0	$c_{m-l-1,k+1}$	$c_{m-l-1,k+2}$	...	$c_{m-l-1,m-l-1}$	0
	0	0	...	0	

Table 1: The c-table

which can be completed easily using Algorithm 3.15.

Algorithm 3.15 ([63]). *[Computing the coefficients* $c_{ij}(m,k,l)$*]*

Input: m, k, l

Output: table of the quantities $c_{ij} \equiv c_{ij}(m,k,l)$ $(i,j = k+1, k+2, \ldots, m-l-1)$

Step 1. *Compute recursively* $c_{k+1,k+1}$, $c_{k+1,k+2}, \ldots, c_{k+1,m-l-1}$ *by the formulas*

$$c_{k+1,m-l-1} := \binom{m}{k+1}^{-1}\binom{m}{l+1}^{-1}\frac{(-1)^{m-k-l-2}(m-k-l-1)_{2k+2l+4}}{(2k+2)!(2l+2)!},$$

$$c_{k+1,j} := \frac{(j-m)(j-k)(j+k+3)}{(j+1)(j-m+l+1)(j-m-l-1)}\,c_{k+1,j+1} \qquad (j = m-l-2, \ldots, k+2, k+1).$$

Step 2. *For* $i = k+1, k+2, \ldots, m-l-2$ *and* $j = k+1, k+2, \ldots, m-l-1$, *compute* $c_{i+1,j}$ *using the recurrence* (3.19).

Observe that the complexity of Algorithm 3.15 is $O(m^2)$.

3.3.2 Computing the inner products $\left\langle B_j^n, D_i^{(m,k,l)}(\cdot;\alpha,\beta)\right\rangle_{\alpha\beta}$

Now, we recall an efficient method of computing the inner products $\phi_{ij} := \left\langle B_j^n, D_i^{(m,k,l)}(\cdot;\alpha,\beta)\right\rangle_{\alpha\beta}$ $(i = k+1, k+2, \ldots, m-l-1;\ j = 0,1,\ldots,n)$. As we shall see in Chapter 4, Algorithm 3.16 plays a significant role in solving Problem 1.9.

According to [64, 103], we have

$$\phi_{ij} := \binom{m-k-l-2}{i-k-1}\binom{m}{i}^{-1}\binom{n}{j}\frac{(\alpha+l+2)_{n-j}(\beta+k+2)_j}{(\alpha+l+2)_{l+1}(\beta+k+2)_{k+1}}\psi_{ij} \qquad (3.20)$$
$$(i=k+1,k+2,\ldots,m-l-1;\ j=0,1,\ldots,n),$$

where the quantities ψ_{ij} can be written in terms of the so-called *dual discrete Bernstein polynomials*. For details, see [103]. From [64], we know that the quantities ψ_{ij} can be put in a rectangular table (see Table 2) and the entries of this ψ-table can be computed using Algorithm 3.16 (cf. [103]).

$$\begin{array}{cccc}
\psi_{k+1,0} & \psi_{k+1,1} & \cdots & \psi_{k+1,n} \\
\psi_{k+2,0} & \psi_{k+2,1} & \cdots & \psi_{k+2,n} \\
\cdots & \cdots & \cdots & \cdots \\
\psi_{m-l-1,0} & \psi_{m-l-1,1} & \cdots & \psi_{m-l-1,n}
\end{array}$$

Table 2: The ψ-table

Algorithm 3.16 ([64]). *[Computing the coefficients ψ_{ij}]*
Input: *n, m, k, l, α, β*
Assumptions: *$n > m > 0$; $k, l \geq -1$; $k+l < m-1$; $\alpha, \beta > -1$*
Output: *table of the quantities ψ_{ij} $(i=k+1,k+2,\ldots,m-l-1;\ j=0,1,\ldots,n)$*

Let $c := k+l+2$, and $a := \alpha+\beta$.

Step 1. *Compute the boundary values $\psi_{k+1,0}, \psi_{k+1,1}, \ldots, \psi_{k+1,n}$ by the formula*

$$\psi_{k+1,j} = \frac{(c-n+1)_{m-c}}{(m-c)!(m+a+c+2)_{n-c}}\sum_{i=0}^{m-c}\frac{(c-m)_i(m+a+c+2)_i(j+l-n+1)_i}{i!(\alpha+2l+3)_i(c-n+1)_i}.$$

Step 2. *For $j=0,n$, compute recursively the auxiliary quantities $q_0(j), q_1(j), \ldots, q_{m-c}(j)$ by*

$$q_{i+1}(j) = F(i)q_i(j) + G(i)q_{i-1}(j) \qquad (i=0,1,\ldots,m-c-1;\ q_0(j)=1, q_{-1}(j)=0),$$

where

$$F(i) := 1 - \frac{(n-l-j-1)(m+i+a+c+2)}{(n-m+i)(\alpha+2l+i+3)} + \frac{i(n+i+a+c+1)}{(n-m+i)(\alpha+2l+i+3)},$$

$$G(i) := \frac{i(n+i+a+c+1)(m-i-j-l)}{(n-m+i-1)_2(\alpha+2l+i+3)}.$$

Step 3. *Compute the boundary values $\psi_{k+2,0}, \psi_{k+3,0}, \ldots, \psi_{m-l-1,0}$ (the first column), and $\psi_{k+2,n}, \psi_{k+3,n}, \ldots, \psi_{m-l-1,n}$ (the last column) by the formula*

$$\psi_{ij} = \frac{(c-n+1)_{m-c}(k-l-\alpha-m)_{i-k-1}}{(\beta+2k+3)_{i-k-1}(m-c)!(m+a+c+2)_{n-c}} \times \sum_{p=0}^{i-k-1}\frac{(k-i+1)_p(-m-n-a-1)_p}{(k-l-\alpha-m)_p(c-n+1)_p}q_{m-c-p}(j).$$

`Step 4.` *For* $i = k+1, k+2, \ldots, m-l-2$, *and* $j = 1, 2, \ldots, n-1$, *compute* $\psi_{i+1,j}$ *by*

$$\psi_{i+1,j} = \frac{A(n,j)\psi_{i,j-1} + [D(m,i) - D(n,j)]\psi_{ij} + C(n,j)\psi_{i,j+1} - A(m,i)\psi_{i-1,j}}{C(m,i)}$$

with $A(r,s) := (k-s+1)(r+l-s+\alpha+2)$, $C(r,s) := (s+l-r+1)(k+s+\beta+2)$, *and* $D(r,s) := A(r,s) + C(r,s)$.

Note that the complexity of Algorithm 3.16 is $O(mn)$.

Further properties of dual Bernstein polynomials can be found, e.g., in [21, 62, 103].

Chapter 4

$G^{k,l}$-constrained degree reduction of Bézier curves

In [103], Woźny and Lewanowicz solved the problem of $C^{k,l}$-constrained degree reduction of Bézier curves (see Problem 1.8), using the properties of dual Bernstein polynomials (see §3.3). Assuming that the input and output curves are of degree n and m, respectively, their method has the least complexity, $O(mn)$, among the existing algorithms. Furthermore, they avoided matrix inversion and explicit basis transformation. Some other methods of dealing with Problem 1.8 are summarized in §1.6.1. In this chapter, we apply an extended version of the method from [103] as an essential part of the algorithms of solving Problem 1.9. A goal is to keep the positive features of the older method. In the thesis, we give a more detailed version of the results published in [44].

The outline of the chapter is as follows. In §4.1, we formulate and solve a certain *model problem of constrained degree reduction of Bézier curves*. §4.2 brings complete solutions of Problem 1.9 with and without the simplifying assumptions, i.e., the $G^{k,l}$-constrained and $C^{p,q}/G^{k,l}$-constrained degree reduction of Bézier curves, respectively (see §1.6.1, §2.2 and §2.3). In order to get explicit formulas for the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$, we focus on selected cases of Problem 1.9 (see §4.3). §4.4 deals with the algorithmic implementation of the proposed methods. Some illustrative examples are given in §4.5.

4.1 Degree reduction of Bézier curves with prescribed boundary control points

Recall that, in §2.2, we have related the $G^{k,l}$ continuity conditions (1.20) with the control points of the curves (1.17) and (1.18). As a result, the control points $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$ depend on the unknown parameters λ_i $(i = 1, 2, \ldots, k)$ and μ_j $(j = 1, 2, \ldots, l)$, respectively. Since the constraints (1.20), for k, $l > 3$, are known to be overly restrictive, we have limited ourselves to the cases of $-1 \leq k$, $l \leq 3$. The continuity parameters $\{\lambda_i\}$, $\{\mu_j\}$ and the remaining control points $r_{k+1}, r_{k+2}, \ldots, r_{m-l-1}$ are to be determined so that the weighted L_2-distance (1.19) is minimized.

To begin with, it is convenient to discuss the following model problem of constrained degree reduction of Bézier curves (cf. Problem 1.9).

Problem 4.1. *[Degree reduction of Bézier curves with prescribed boundary control points] Given a Bézier curve $P \in \Pi_n^d$,*

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t),$$

we look for a Bézier curve $R \in \Pi_m^d$ $(m < n)$,

$$R(t) := \sum_{i=0}^{m} r_i B_i^m(t), \tag{4.1}$$

having the prescribed control points $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$, that gives minimum value of the weighted L_2-error

$$E_2^{(\alpha,\beta)} := \|P - R\|_{L_2}^{(\alpha,\beta)} = \sqrt{\int_0^1 (1-t)^\alpha t^\beta \|P(t) - R(t)\|^2 \,\mathrm{d}t} \qquad (\alpha,\, \beta > -1). \tag{4.2}$$

Given the points $p_i := (p_{i1}, p_{i2}, \ldots, p_{id}) \in \mathbb{R}^d$ $(i = 0, 1, \ldots, n)$ and $r_i := (r_{i1}, r_{i2}, \ldots, r_{id}) \in \mathbb{R}^d$ $(i = 0, 1, \ldots, m)$, we use notation $\mathbf{p}^h$, $\mathbf{r}^h$ for the vectors of hth coordinates of the points $p_0, p_1, \ldots, p_n$ and $r_0, r_1, \ldots, r_m$, respectively,

$$\mathbf{p}^h := [p_{0h}, p_{1h}, \ldots, p_{nh}], \qquad \mathbf{r}^h := [r_{0h}, r_{1h}, \ldots, r_{mh}] \qquad (h = 1, 2, \ldots, d).$$

As an extension of the result given in [103] (see also [64]), we obtain the following theorem.

Theorem 4.2. *The* inner *control points $r_i = (r_{i1}, r_{i2}, \ldots, r_{id})$ $(k+1 \le i \le m-l-1)$ of the curve* (4.1)*, being the solution of Problem 4.1, are given by*

$$r_i = \sum_{j=0}^{n} \upsilon_j \phi_{ij} \qquad (i = k+1, k+2, \ldots, m-l-1), \tag{4.3}$$

where

$$\phi_{ij} = \left\langle B_j^n, D_i^{(m,k,l)}(\cdot;\alpha,\beta) \right\rangle_{\alpha\beta}$$

(see (3.16) *and §3.3.2),*

$$\upsilon_j := p_j - \binom{n}{j}^{-1} \left(\sum_{h=0}^{k} + \sum_{h=m-l}^{m} \right) \binom{n-m}{j-h} \binom{m}{h} r_h \qquad (j = 0, 1, \ldots, n). \tag{4.4}$$

The weighted L_2-error (4.2) *is given by*

$$E_2^{(\alpha,\beta)} = \sqrt{\sum_{h=1}^{d} [I_{nn}(\mathbf{p}^h, \mathbf{p}^h) + I_{mm}(\mathbf{r}^h, \mathbf{r}^h) - 2I_{nm}(\mathbf{p}^h, \mathbf{r}^h)]}, \tag{4.5}$$

where for $\mathbf{a} := [a_0, a_1, \ldots, a_N]$ and $\mathbf{b} := [b_0, b_1, \ldots, b_M]$, we define

$$I_{NM}(\mathbf{a}, \mathbf{b}) := \frac{\mathrm{B}(\alpha+1, \beta+1)}{(\alpha+\beta+2)_{N+M}} \sum_{i=0}^{N} \sum_{j=0}^{M} \binom{N}{i} \binom{M}{j} (\alpha+1)_{N+M-i-j} (\beta+1)_{i+j} a_i b_j,$$

where $\mathrm{B}(\alpha, \beta) := \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ *is called* beta function.

Proof. To begin with, we write

$$R(t) = S(t) + T(t),$$

where

$$S(t) := \sum_{i=k+1}^{m-l-1} r_i B_i^m(t), \qquad T(t) := \left(\sum_{i=0}^{k} + \sum_{i=m-l}^{m}\right) r_i B_i^m(t).$$

Using the degree elevation formula (see, e.g., [33, §6.10]; we adopt the usual convention that $\binom{u}{v} = 0$ if $v < 0$ or $v > u$)

$$B_i^m(t) = \binom{m}{i} \sum_{h=0}^{n} \binom{n-m}{h-i} \binom{n}{h}^{-1} B_h^n(t),$$

we obtain

$$T(t) = \sum_{j=0}^{n} d_j B_j^n(t),$$

where

$$d_j := \binom{n}{j}^{-1} \left(\sum_{h=0}^{k} + \sum_{h=m-l}^{m}\right) \binom{n-m}{j-h} \binom{m}{h} r_h.$$

Now, we observe that

$$\|P - R\|_{L_2}^{(\alpha,\beta)} = \|W - S\|_{L_2}^{(\alpha,\beta)} = \sqrt{\sum_{h=1}^{d} \int_0^1 (1-t)^\alpha t^\beta \left[W^h(t) - S^h(t)\right]^2 \mathrm{d}t},$$

where

$$W(t) := \left[W^1(t), W^2(t), \ldots, W^d(t)\right] = P(t) - T(t) = \sum_{i=0}^{n} \upsilon_i B_i^n(t),$$
$$S(t) := \left[S^1(t), S^2(t), \ldots, S^d(t)\right]$$

with

$$\upsilon_i := p_i - d_i.$$

Thus, we are looking for the best weighted least squares approximation for W^h $(h = 1, 2, \ldots, d)$ in the space $\Pi_m^{(k,l)}$ (see (3.17)). Remembering that B_i^m and $D_i^{(m,k,l)}(\cdot;\alpha,\beta)$ $(k+1 \le i \le m-l-1)$ are the dual bases in the space $\Pi_m^{(k,l)}$, we use Fact 3.2 and obtain

$$r_i = \sum_{j=0}^{n} \upsilon_j \left\langle B_j^n, D_i^{(m,k,l)}(\cdot;\alpha,\beta)\right\rangle_{\alpha\beta} = \sum_{j=0}^{n} \upsilon_j \phi_{ij} \qquad (i = k+1, k+2, \ldots, m-l-1),$$

which is the formula (4.3).

It can be easily checked that for

$$P(t) := \left[P^1(t), P^2(t), \ldots, P^d(t)\right], \qquad R(t) := \left[R^1(t), R^2(t), \ldots, R^d(t)\right],$$

we have

$$I_{nn}(\mathbf{p}^h,\mathbf{p}^h)=\left(\|P^h\|_{L_2}^{(\alpha,\beta)}\right)^2,\qquad I_{mm}(\mathbf{r}^h,\mathbf{r}^h)=\left(\|R^h\|_{L_2}^{(\alpha,\beta)}\right)^2,$$
$$I_{nm}(\mathbf{p}^h,\mathbf{r}^h)=\left\langle P^h,R^h\right\rangle_{\alpha\beta}.$$

Hence, the formula (4.5) follows. □

4.2 Computing the continuity parameters

Coming back to the problem of $G^{k,l}$-constrained degree reduction of Bézier curves (see Problem 1.9), we notice that the formulas (2.11)–(2.16) with *fixed parameters* $\{\lambda_i\}$ and $\{\mu_j\}$ (cf. §2.2) constitute constraints of the form demanded in Problem 4.1. As a result, the control points (4.3) depend on these parameters.

Now, *optimum values* of the parameters can be obtained by minimizing the squared error (4.5),

$$E^{(\alpha,\beta)}\equiv E^{(\alpha,\beta)}(\lambda_1,\lambda_2,\ldots,\lambda_k,\mu_1,\mu_2,\ldots,\mu_l):=\left(E_2^{(\alpha,\beta)}\right)^2, \tag{4.6}$$

depending on $\{\lambda_i\}$ and $\{\mu_j\}$ *via* formulas (2.11)–(2.16) and (4.3). For a minimum of the function (4.6), it is necessary that its derivatives with respect to the parameters are zero, which yields the system

$$\left.\begin{aligned}\frac{\partial}{\partial\lambda_u}E^{(\alpha,\beta)}&=\sum_{h=1}^{d}\left[\frac{\partial}{\partial\lambda_u}I_{mm}(\mathbf{r}^h,\mathbf{r}^h)-2\frac{\partial}{\partial\lambda_u}I_{nm}(\mathbf{p}^h,\mathbf{r}^h)\right]=0 \qquad (u=1,2,\ldots,k),\\ \frac{\partial}{\partial\mu_v}E^{(\alpha,\beta)}&=\sum_{h=1}^{d}\left[\frac{\partial}{\partial\mu_v}I_{mm}(\mathbf{r}^h,\mathbf{r}^h)-2\frac{\partial}{\partial\mu_v}I_{nm}(\mathbf{p}^h,\mathbf{r}^h)\right]=0 \qquad (v=1,2,\ldots,l).\end{aligned}\right\} \tag{4.7}$$

Using the notation

$$H:=\frac{\mathrm{B}(\alpha+1,\beta+1)}{(\alpha+\beta+2)_m}, \tag{4.8}$$

$$F_{tj}(\mathbf{q}):=\frac{1}{(\alpha+\beta+m+2)_t}\binom{m}{j}\sum_{i=0}^{t}\binom{t}{i}(\alpha+1)_{t+m-i-j}(\beta+1)_{i+j}q_i, \tag{4.9}$$

where $\mathbf{q}=[q_0,q_1,\ldots,q_t]$, we obtain

$$\frac{\partial}{\partial\lambda_u}I_{nm}(\mathbf{p}^h,\mathbf{r}^h)=H\sum_{j=u}^{m-l-1}F_{nj}(\mathbf{p}^h)\frac{\partial r_{jh}}{\partial\lambda_u},$$

$$\frac{\partial}{\partial\mu_v}I_{nm}(\mathbf{p}^h,\mathbf{r}^h)=H\sum_{j=k+1}^{m-v}F_{nj}(\mathbf{p}^h)\frac{\partial r_{jh}}{\partial\mu_v},$$

and

$$\frac{\partial}{\partial\lambda_u}I_{mm}(\mathbf{r}^h,\mathbf{r}^h)=\frac{H}{(\alpha+\beta+m+2)_m}\sum_{i=0}^{m}\sum_{j=0}^{m}\binom{m}{i}\binom{m}{j}(\alpha+1)_{2m-i-j}(\beta+1)_{i+j}$$
$$\times\left[\frac{\partial r_{ih}}{\partial\lambda_u}r_{jh}+r_{ih}\frac{\partial r_{jh}}{\partial\lambda_u}\right]=2H\sum_{j=u}^{m-l-1}F_{mj}(\mathbf{r}^h)\frac{\partial r_{jh}}{\partial\lambda_u},$$

$$\frac{\partial}{\partial\mu_v}I_{mm}(\mathbf{r}^h,\mathbf{r}^h)=2H\sum_{j=k+1}^{m-v}F_{mj}(\mathbf{r}^h)\frac{\partial r_{jh}}{\partial\mu_v}.$$

Hence, the system (4.7) takes the form

$$\left.\begin{aligned}\sum_{h=1}^{d}\sum_{j=u}^{m-l-1}\left[F_{mj}(\mathbf{r}^h)-F_{nj}(\mathbf{p}^h)\right]\frac{\partial r_{jh}}{\partial\lambda_u}=0 \qquad & (u=1,2,\ldots,k),\\ \sum_{h=1}^{d}\sum_{j=k+1}^{m-v}\left[F_{mj}(\mathbf{r}^h)-F_{nj}(\mathbf{p}^h)\right]\frac{\partial r_{jh}}{\partial\mu_v}=0 \qquad & (v=1,2,\ldots,l).\end{aligned}\right\} \tag{4.10}$$

In the case of $k=l=3$, we compute the partial derivatives of hth coordinates of the control points (2.11)–(2.16). We obtain the following formulas:

$$\frac{\partial r_{ih}}{\partial\lambda_1}=\begin{cases}\frac{n}{m}\Delta p_{0h} & (i=1),\\ 2\frac{n}{m}\Delta p_{0h}+2\lambda_1\frac{(n-1)_2}{(m-1)_2}\Delta^2p_{0h} & (i=2),\\ 3\frac{n}{m}\Delta p_{0h}+\left[2\lambda_1+\frac{1}{m-2}\lambda_2\right]3\frac{(n-1)_2}{(m-1)_2}\Delta^2p_{0h}+3\lambda_1^2\frac{(n-2)_3}{(m-2)_3}\Delta^3p_{0h} & (i=3),\\ 0 & (i=0;\ m-3\le i\le m),\end{cases} \tag{4.11}$$

$$\frac{\partial r_{ih}}{\partial\lambda_2}=\begin{cases}\frac{n}{(m-1)_2}\Delta p_{0h} & (i=2),\\ 3\frac{n}{(m-1)_2}\Delta p_{0h}+3\lambda_1\frac{(n-1)_2}{(m-2)_3}\Delta^2p_{0h} & (i=3),\\ 0 & (i=0,1;\ m-3\le i\le m),\end{cases} \tag{4.12}$$

$$\frac{\partial r_{ih}}{\partial\lambda_3}=\begin{cases}\frac{n}{(m-2)_3}\Delta p_{0h} & (i=3),\\ 0 & (i=0,1,2;\ m-3\le i\le m),\end{cases} \tag{4.13}$$

$$\frac{\partial r_{ih}}{\partial\mu_1}=\begin{cases}-\frac{n}{m}\Delta p_{n-1,h} & (i=m-1),\\ -2\frac{n}{m}\Delta p_{n-1,h}+2\mu_1\frac{(n-1)_2}{(m-1)_2}\Delta^2p_{n-2,h} & (i=m-2),\\ -3\frac{n}{m}\Delta p_{n-1,h}+\left[2\mu_1-\frac{1}{m-2}\mu_2\right]3\frac{(n-1)_2}{(m-1)_2}\Delta^2p_{n-2,h}-3\mu_1^2\frac{(n-2)_3}{(m-2)_3}\Delta^3p_{n-3,h} & (i=m-3),\\ 0 & (0\le i\le 3;\ i=m),\end{cases} \tag{4.14}$$

$$\frac{\partial r_{ih}}{\partial\mu_2}=\begin{cases}\frac{n}{(m-1)_2}\Delta p_{n-1,h} & (i=m-2),\\ 3\frac{n}{(m-1)_2}\Delta p_{n-1,h}-3\mu_1\frac{(n-1)_2}{(m-2)_3}\Delta^2p_{n-2,h} & (i=m-3),\\ 0 & (0\le i\le 3;\ i=m-1,m),\end{cases} \tag{4.15}$$

$$\frac{\partial r_{ih}}{\partial\mu_3}=\begin{cases}-\frac{n}{(m-2)_3}\Delta p_{n-1,h} & (i=m-3),\\ 0 & (0\le i\le 3;\ m-2\le i\le m).\end{cases} \tag{4.16}$$

Notice that the partial derivatives of hth coordinates of the control points (4.3) depend on (4.11)–(4.16) in the following way:

$$\frac{\partial r_{ih}}{\partial \lambda_u} = -\sum_{j=0}^{n} \binom{n}{j}^{-1} \sum_{g=u}^{k} \binom{n-m}{j-g} \binom{m}{g} \phi_{ij} \frac{\partial r_{gh}}{\partial \lambda_u}, \tag{4.17}$$

$$\frac{\partial r_{ih}}{\partial \mu_v} = -\sum_{j=0}^{n} \binom{n}{j}^{-1} \sum_{g=m-l}^{m-v} \binom{n-m}{j-g} \binom{m}{g} \phi_{ij} \frac{\partial r_{gh}}{\partial \mu_v}. \tag{4.18}$$

One can easily see that when $k, l \leq 3$, we compute $\frac{\partial r_{ih}}{\partial \lambda_u}$, $\frac{\partial r_{ih}}{\partial \mu_v}$ by (4.17), (4.18) if $k < i < m - l$, and by (4.11)–(4.16) otherwise. Finally, we put the expressions (4.11)–(4.18) into the equations of the system (4.10).

Remark 4.3. Observe that for $k \geq 2$ or $l \geq 2$, the system (4.10) is nonlinear, therefore, quite difficult to solve. Moreover, from a practical point of view, we additionally require that λ_1, $\mu_1 > 0$, which results in the same directions of tangent vectors at the endpoints of the curves (1.17) and (1.18). Thus, to guarantee that these conditions will be satisfied, it is not enough just to solve the system (4.10).

Now, we discuss two possible ways of computing $\{\lambda_i\}$ and $\{\mu_j\}$.

4.2.1 Computing $G^{k,l}$ parameters using quadratic and nonlinear programming approach

It is easy to check that if ($k = 1$ and $l \leq k$) or ($l = 1$ and $k \leq l$), then the error (4.6) is a quadratic function of the continuity parameters.

In the case of ($k = 2$ and $l \leq k$) or ($l = 2$ and $k \leq l$), the error (4.6) is a fourth-degree polynomial function of the continuity parameters.

For ($k = 3$ and $l \leq k$) or ($l = 3$ and $k \leq l$), the error (4.6) is a sixth-degree polynomial function of the continuity parameters.

To find optimum values of the parameters λ_1, μ_1 in the case of $G^{1,1}$-constrained degree reduction problem, assuming that α, $\beta = 0$, Lu and Wang [77] solved the quadratic programming problem subject to the constraints

$$\lambda_1 \geq z_0, \quad \mu_1 \geq z_1, \tag{4.19}$$

where z_0 and z_1 are positive lower bounds prescribed to small values (they set 10^{-4} for both lower bounds in the examples section). Such an approach can be used in the cases which result in a quadratic error function (4.6). One can solve the quadratic programming problem using, e.g., an *iterative active set method*, which is implemented in many software libraries. The active set mechanism used by standard quadratic solvers is described in [14, §6.5].

Analogously, one can observe that for $k = 2, 3$ or $l = 2, 3$, the problem of minimizing the error (4.6) subject to the constraints (4.19) is a nonlinear programming problem. To solve it, one can use, for instance, a *sequential quadratic programming (SQP) method* (see, e.g., [14, §15.1]), which is also widely available.

4.2.2 Computing $C^{p,q}/G^{k,l}$ parameters by solving a system of linear equations

In the $G^{2,2}$-constrained case, Rababah and Mann [91] simplified the problem by considering $C^{1,1}$ continuity at the endpoints, i.e., they set λ_1, $\mu_1 := 1$. Later, this approach was also used by Lu [70]. In [92], the same idea was used to simplify the $G^{3,3}$-constrained case and the authors noted that such an approach leads to a system of linear equations.

Now, using the notation from §2.3, we generalize the above-described approach for any k, l such that $-1 \leq k,\, l \leq 3$. If $k \geq 2$, we set $\lambda_1 := 1$, which implies C^1 continuity at $t = 0$ and consequently, $G^{k,l}$ constraints become $C^{1,q}/G^{k,l}$ constraints, where $q \in \{-, 1\}$. Similarly, when $l \geq 2$, we set $\mu_1 := 1$, which implies C^1 continuity at $t = 1$ and consequently, $G^{k,l}$ constraints become $C^{p,1}/G^{k,l}$ constraints, where $p \in \{-, 1\}$.

Note that in the cases of $k = 2, 3$ or $l = 2, 3$, the above-described method leads to the linear system (4.10) and the error (4.6) is a quadratic function of the continuity parameters. However, in the cases of $k = 1$ or $l = 1$, there is no guarantee that the solution satisfies $\lambda_1 > 0$ or $\mu_1 > 0$, respectively. In the case of the solution with nonpositive values of the parameters, we must solve the quadratic programming problem subject to the constraints with prescribed positive lower bounds for the parameters (see (4.19)). Observe that this approach uses no simplifying assumptions for k, $l \leq 1$.

Remark 4.4. Taking into account that the mentioned linear systems are rather small, one can get explicit formulas for the continuity parameters. For example, in the cases of $G^{1,1}$, $C^{-,1}/G^{1,2}$, $C^{1,-}/G^{2,1}$ and $C^{1,1}/G^{2,2}$, the linear system (4.10) has the following form:

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1,\\ a_{21}x_1 + a_{22}x_2 &= b_2, \end{aligned}\right\} \tag{4.20}$$

where x_1, x_2 are the continuity parameters and the coefficients a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2 depend on a considered case. Explicit formulas for these coefficients are given in §4.3. Using the notation of (4.20), we obtain the following solution:

$$x_1 = \frac{b_1a_{22} - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}}, \qquad x_2 = \frac{a_{11}b_2 - b_1a_{21}}{a_{11}a_{22} - a_{12}a_{21}}. \tag{4.21}$$

Most of the known algorithms solve a system of normal equations in order to get expressions for the inner control points (4.3). Such an approach makes those expressions dependent on the inverse of a certain matrix. As a result, formulas for the continuity parameters also depend on the inverse (see, e.g., [70, 114]). Since the method given in this chapter is based on Theorem 4.2, the formulas are truly explicit.

Observe that in the above-mentioned cases, one can easily give the following form of the quadratic error function (4.6):

$$E^{(\alpha,\beta)}(x_1, x_2) = \frac{1}{2}\begin{bmatrix} x_1, & x_2 \end{bmatrix}\begin{bmatrix} a_{11}^* & a_{12}^* \\ a_{21}^* & a_{22}^* \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1, & x_2 \end{bmatrix}\begin{bmatrix} b_1^* \\ b_2^* \end{bmatrix} + c^* = E_*^{(\alpha,\beta)}(x_1, x_2) + c^*, \tag{4.22}$$

where

$$a_{ij}^* := 2Ha_{ij},\ b_i^* := -2Hb_i \quad (i, j = 1, 2) \tag{4.23}$$

for (4.8). Obviously, a constant c^* is meaningless in the minimization process, therefore, $E_*^{(\alpha,\beta)}(x_1, x_2)$ denotes the significant terms of $E^{(\alpha,\beta)}(x_1, x_2)$. If the solution (4.21) does not

fulfill the conditions λ_1, $\mu_1 > 0$, then one can use $E_*^{(\alpha,\beta)}(x_1, x_2)$ as an objective function for the minimization problem with constraints.

4.3 Explicit formulas for the continuity parameters

In Remark 4.4, we describe the method of computing the parameters $\{\lambda_i\}$ and $\{\mu_j\}$ in the cases of $G^{1,1}$, $C^{-,1}/G^{1,2}$, $C^{1,-}/G^{2,1}$ and $C^{1,1}/G^{2,2}$. These parameters depend on certain coefficients (see (4.21)). Now, we give explicit formulas for the coefficients.

4.3.1 $G^{1,1}$-constrained case

In the case of $G^{1,1}$ continuity conditions, we set $x_1 := \lambda_1$, $x_2 := \mu_1$, and compute the required coefficients by

$$
\begin{aligned}
&a_{11} := A_1^{(0,0)}(1, m-2, 1, 0, 1, m-2, 1, 0),\\
&a_{12}, a_{21} := -A_1^{(0,0)}(1, m-2, 1, 0, 2, m-1, m-1, n-1),\\
&a_{22} := A_1^{(0,0)}(2, m-1, m-1, n-1, 2, m-1, m-1, n-1),\\
&b_1 := -A_2^{(0,0)}(1, m-2, 1, 0),\\
&b_2 := A_2^{(0,0)}(2, m-1, m-1, n-1),
\end{aligned}
$$

where

$$
A_1^{(u,v)}(a, b, c, e, f, g, s, t) := \sum_{h=1}^{d}\sum_{j=a}^{b} N_u(j, \mathbf{p}^h, c, e)\sum_{i=f}^{g} L(i, j)N_v(i, \mathbf{p}^h, s, t), \tag{4.24}
$$

$$
A_2^{(u,v)}(a, b, c, e) := \sum_{h=1}^{d}\sum_{j=a}^{b} N_u(j, \mathbf{p}^h, c, e)\left[\sum_{i=0}^{m} L(i, j)K_v(i, \mathbf{p}^h) - F_{nj}(\mathbf{p}^h)\right] \tag{4.25}
$$

with (4.9),

$$
L(i, j) := (\alpha+\beta+m+2)_m^{-1}\binom{m}{j}\binom{m}{i}(\alpha+1)_{2m-i-j}(\beta+1)_{i+j}, \tag{4.26}
$$

$$
N_0(j, \mathbf{q}, s, t) := \begin{cases} \dfrac{n}{m}\Delta q_t & (j = s),\\[2ex] -n\Delta q_t \displaystyle\sum_{i=0}^{n}\binom{n}{i}^{-1}\binom{n-m}{i-s}\phi_{ji} & \text{otherwise,}\end{cases} \tag{4.27}
$$

$$
K_0(i, \mathbf{q}) := \begin{cases} q_0 & (i = 0, 1),\\[2ex] \displaystyle\sum_{j=0}^{n}\phi_{ij}\left[q_j - \binom{n}{j}^{-1}\left(\sum_{h=0}^{1} + \sum_{h=m-1}^{m}\right)M(j, h)K_0(h, \mathbf{q})\right] & (2 \le i \le m-2),\\[2ex] q_n & (i = m-1, m)\end{cases}
$$

for

$$\mathbf{q} := [q_0, q_1, \ldots, q_n], \qquad M(i,j) := \binom{n-m}{i-j}\binom{m}{j}. \tag{4.28}$$

4.3.2 $C^{1,1}/G^{2,2}$-constrained case

In the case of $C^{1,1}/G^{2,2}$ continuity conditions, we set λ_1, $\mu_1 := 1$, $x_1 := \lambda_2$, $x_2 := \mu_2$, and compute the required coefficients by

$$\begin{aligned}
a_{11} &:= A_1^{(1,1)}(2, m-3, 2, 0, 2, m-3, 2, 0),\\
a_{12}, a_{21} &:= A_1^{(1,1)}(2, m-3, 2, 0, 3, m-2, m-2, n-1),\\
a_{22} &:= A_1^{(1,1)}(3, m-2, m-2, n-1, 3, m-2, m-2, n-1),\\
b_1 &:= -A_2^{(1,1)}(2, m-3, 2, 0),\\
b_2 &:= -A_2^{(1,1)}(3, m-2, m-2, n-1),
\end{aligned}$$

using (4.24) and (4.25) with (4.9), (4.26),

$$N_1(j, \mathbf{q}, s, t) := \begin{cases} \dfrac{1}{m-1} N_0(j, \mathbf{q}, s, t) & (j = s),\\ \dfrac{1}{(m-1)_2}\binom{m}{s} N_0(j, \mathbf{q}, s, t) & \text{otherwise,} \end{cases} \tag{4.29}$$

$$K_1(i, \mathbf{q}) := \begin{cases} q_0 & (i = 0),\\ q_0 + \dfrac{n}{m}\Delta q_0 & (i = 1),\\ q_0 + 2\dfrac{n}{m}\Delta q_0 + \dfrac{(n-1)_2}{(m-1)_2}\Delta^2 q_0 & (i = 2),\\ \displaystyle\sum_{j=0}^{n} \phi_{ij}\left[q_j - \binom{n}{j}^{-1}\left(\sum_{h=0}^{2} + \sum_{h=m-2}^{m}\right) M(j,h) K_1(h, \mathbf{q})\right] & (3 \le i \le m-3),\\ q_n - 2\dfrac{n}{m}\Delta q_{n-1} + \dfrac{(n-1)_2}{(m-1)_2}\Delta^2 q_{n-2} & (i = m-2),\\ q_n - \dfrac{n}{m}\Delta q_{n-1} & (i = m-1),\\ q_n & (i = m) \end{cases} \tag{4.30}$$

for (4.27) and (4.28).

4.3.3 $C^{1,-}/G^{2,1}$-constrained case

In the case of $C^{1,-}/G^{2,1}$ continuity conditions, we set $\lambda_1 := 1$, $x_1 := \lambda_2$, $x_2 := \mu_1$, and compute the required coefficients by

$$\begin{aligned}
a_{11} &:= A_1^{(1,1)}(2, m-2, 2, 0, 2, m-2, 2, 0),\\
a_{12}, a_{21} &:= -A_1^{(0,1)}(3, m-1, m-1, n-1, 2, m-2, 2, 0),\\
a_{22} &:= A_1^{(0,0)}(3, m-1, m-1, n-1, 3, m-1, m-1, n-1),\\
b_1 &:= -A_2^{(1,2)}(2, m-2, 2, 0),\\
b_2 &:= A_2^{(0,2)}(3, m-1, m-1, n-1),
\end{aligned}$$

using (4.24) and (4.25) with (4.9), (4.26), (4.27), (4.29) and

$$K_2(i, \mathbf{q}) := \begin{cases} K_1(i, \mathbf{q}) & (i = 0, 1, 2),\\ \displaystyle\sum_{j=0}^{n} \phi_{ij} \left[q_j - \binom{n}{j}^{-1} \left(\sum_{h=0}^{2} + \sum_{h=m-1}^{m} \right) M(j,h) K_2(h, \mathbf{q}) \right] & (3 \le i \le m-2),\\ K_1(m, \mathbf{q}) & (i = m-1, m) \end{cases}$$

for (4.28) and (4.30).

4.3.4 $C^{-,1}/G^{1,2}$-constrained case

In the case of $C^{-,1}/G^{1,2}$ continuity conditions, we set $\mu_1 := 1$, $x_1 := \lambda_1$, $x_2 := \mu_2$, and compute the required coefficients by

$$\begin{aligned}
a_{11} &:= A_1^{(0,0)}(1, m-3, 1, 0, 1, m-3, 1, 0),\\
a_{12}, a_{21} &:= A_1^{(0,1)}(1, m-3, 1, 0, 2, m-2, m-2, n-1),\\
a_{22} &:= A_1^{(1,1)}(2, m-2, m-2, n-1, 2, m-2, m-2, n-1),\\
b_1 &:= -A_2^{(0,3)}(1, m-3, 1, 0),\\
b_2 &:= -A_2^{(1,3)}(2, m-2, m-2, n-1),
\end{aligned}$$

using (4.24) and (4.25) with (4.9), (4.26), (4.27), (4.29) and

$$K_3(i, \mathbf{q}) := \begin{cases} K_1(0, \mathbf{q}) & (i = 0, 1),\\ \displaystyle\sum_{j=0}^{n} \phi_{ij} \left[q_j - \binom{n}{j}^{-1} \left(\sum_{h=0}^{1} + \sum_{h=m-2}^{m} \right) M(j,h) K_3(h, \mathbf{q}) \right] & (2 \le i \le m-3),\\ K_1(i, \mathbf{q}) & (i = m-2, m-1, m) \end{cases}$$

for (4.28) and (4.30).

4.4 Algorithms

In this subsection, we show the details of implementation of the proposed methods of $C^{p,q}/G^{k,l}$-constrained and $G^{k,l}$-constrained degree reduction of Bézier curves.

4.4.1 Auxiliary computations

To begin with, recall that the quantities ϕ_{ij} $(i = k+1, k+2, \ldots, m-l-1;\ j = 0, 1, \ldots, n)$ are related to the coefficients ψ_{ij} (see (3.20)). The latter can be put in a rectangular table (see Table 2) and computed using Algorithm 3.16.

Next, we use Theorem 4.2 to give the algorithm of computing expressions for the control points of the curve (1.18).

Algorithm 4.5. *[Evaluation of the control points]*

Input: α, β – *parameters of the weighted* L_2*-norm* (1.13)*;*

n, $p_0, p_1, \ldots, p_n$ – *degree and control points of the original Bézier curve* (1.17)*;*

m – *degree of the reduced Bézier curve* (1.18)*;*

k, l – *orders of the geometric continuity (see* (1.20)*);*

ϕ*-table precomputed using Algorithm 3.16 and the formula* (3.20)

Optional input: *values of the continuity parameters* $\lambda_1, \lambda_2, \ldots, \lambda_k$ *and* $\mu_1, \mu_2, \ldots, \mu_l$

Assumptions: $n > m > 0;\ -1 \le k, l \le 3;\ k + l < m - 1;\ \alpha, \beta > -1$

Output: *expressions (or values) for the control points of the Bézier curve* (1.18)

`Step 1.` *Compute*

(i) $r_0, r_1, \ldots, r_k$ *by* (2.11)–(2.13)*;*

(ii) $r_m, r_{m-1}, \ldots, r_{m-l}$ *by* (2.14)–(2.16).

`Step 2.` *Compute* $\upsilon_0, \upsilon_1, \ldots, \upsilon_n$ *by* (4.4).

`Step 3.` *Compute* $r_{k+1}, r_{k+2}, \ldots, r_{m-l-1}$ *by* (4.3).

`Step 4.` *Return* $r_0, r_1, \ldots, r_m$.

Observe that Algorithm 4.5 can be used in two ways. For given values of the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$, we obtain values of the control points $r_0, r_1, \ldots, r_m$. In the case of unknown values for the parameters, the algorithm gives expressions which depend on these parameters. Note that the complexity of Algorithm 4.5 is $O(mn)$.

4.4.2 $C^{p,q}/G^{k,l}$-constrained degree reduction algorithms

Now, we give two algorithms of $C^{p,q}/G^{k,l}$-constrained degree reduction (see §4.2.2). The first algorithm makes use of Remark 4.4, therefore, it solves the cases of $G^{1,1}$, $C^{-,1}/G^{1,2}$, $C^{1,-}/G^{2,1}$ and $C^{1,1}/G^{2,2}$, using explicit formulas for the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$ (see §4.3). However, in the case on nonpositive values for λ_1 or μ_1, the algorithm solves the quadratic programming problem subject to the conditions (4.19). More than 40 different tests were performed (the results of some of them are presented in §4.5). None of them caused such a problem, therefore, we conclude that it happens very rarely. The second algorithm is much

more general. It accepts any k and l not exceeding 3. However, there are no explicit formulas for the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$.

Both algorithms consist of two phases. During Phase A, we minimize the error (4.6), which—by the results given in Theorem 4.2—depends only on the parameters $\{\lambda_i\}$ and $\{\mu_j\}$. As a result, we obtain optimum values for these parameters. During Phase B, we use the obtained values of the continuity parameters to compute the control points $r_0, r_1, \ldots, r_m$ using Algorithm 4.5.

Remark 4.6. According to Remark 2.2, if $-1 \leq k, l < 1$, then we are dealing with the $C^{k,l}$-constrained case. Thus, there are no continuity parameters to determine, and Phase A can be omitted (see Algorithm 4.7, Step 2).

Algorithm 4.7. *[$C^{p,q}/G^{k,l}$-constrained degree reduction of Bézier curves — selected cases]*

Input: α, β – *parameters of the weighted* L_2*-norm* (1.13)*;*
n, $p_0, p_1, \ldots, p_n$ – *degree and control points of the original Bézier curve* (1.17)*;*
m – *degree of the reduced Bézier curve* (1.18)*;*
k, l – *orders of the geometric continuity (see* (1.20)*);*
z_0, z_1 – *lower bounds for the parameters* λ_1 *and* μ_1*, respectively (see* (4.19)*)*

Assumptions: $n > m > 0$; $z_0, z_1 > 0$; $\alpha, \beta > -1$; $k + l < m - 1$; *(*$k = 1$ *and* $l = 2$*) or (*$k = 2$ *and* $l = 1$*) or* $(k = l = 1)$ *or* $(k = l = 2)$ *or* $(-1 \leq k, l < 1)$

Output: control points of the $C^{p,q}/G^{k,l}$*-constrained degree reduced Bézier curve*

Phase A

Step 1. *Compute* ϕ_{ij} $(i = k+1, k+2, \ldots, m-l-1;\ j = 0, 1, \ldots, n)$ *using Algorithm 3.16 and the formula* (3.20).

Step 2. *Check if the remaining steps of Phase A are necessary*

If $(k, l < 1)$ ***then*** *go to* Step 6.

Step 3. *Simplify the problem*

(i) ***If*** $(k > 1)$ ***then*** $\lambda_1 := 1$*;*

(ii) ***If*** $(l > 1)$ ***then*** $\mu_1 := 1$.

Step 4. *Use explicit formulas for the continuity parameters*

(i) ***If*** $(k = l = 1)$ ***then***

- *compute* λ_1 *and* μ_1 *by* (4.21) *using explicit formulas given in §4.3.1;*

(ii) ***If*** $(k = 2$ *and* $l = 1)$ ***then***

- *compute* λ_2 *and* μ_1 *by* (4.21) *using explicit formulas given in §4.3.3;*

(iii) ***If*** $(k = 1$ *and* $l = 2)$ ***then***

- *compute* λ_1 *and* μ_2 *by* (4.21) *using explicit formulas given in §4.3.4;*

(iv) ***If*** $(k = l = 2)$ ***then***

- *compute* λ_2 *and* μ_2 *by* (4.21) *using explicit formulas given in §4.3.2;*
- *go to* Step 6.

`Step 5.` *Check if the solution is feasible*

If *($\lambda_1 \leq 0$ or $\mu_1 \leq 0$)* ***then***

(i) $c := \{\lambda_1 \geq z_0,\ \mu_1 \geq z_1\}$;

(ii) ***If*** *($k \neq 1$)* ***then*** $c := c \setminus \{\lambda_1 \geq z_0\}$;

(iii) ***If*** *($l \neq 1$)* ***then*** $c := c \setminus \{\mu_1 \geq z_1\}$;

(iv) *compute* $a_{11}^*, a_{12}^* = a_{21}^*, a_{22}^*, b_1^*, b_2^*$ *by* (4.23);

(v) *compute values of the continuity parameters by solving the quadratic programming problem of minimizing the error* $E_*^{(\alpha,\beta)}(x_1, x_2)$, *given by* (4.22), *subject to the constraints* c.

Phase B

`Step 6.` *Execute Algorithm 4.5 with the computed values of the continuity parameters, and return the solution, i.e., the control points* $r_0, r_1, \ldots, r_m$.

Algorithm 4.8. *[$C^{p,q}/G^{k,l}$-constrained degree reduction of Bézier curves]*

Input: α, β – *parameters of the weighted* L_2*-norm* (1.13);

n, $p_0, p_1, \ldots, p_n$ – *degree and control points of the original Bézier curve* (1.17);

m – *degree of the reduced Bézier curve* (1.18);

k, l – *orders of the geometric continuity (see* (1.20)*);*

z_0, z_1 – *lower bounds for the parameters* λ_1 *and* μ_1, *respectively (see* (4.19)*)*

Assumptions: $n > m > 0$; $z_0, z_1 > 0$; $\alpha, \beta > -1$; $k + l < m - 1$; $-1 \leq k, l < 3$

Output: *control points of the* $C^{p,q}/G^{k,l}$*-constrained degree reduced Bézier curve*

Phase A

`Step 1.` *Check if the considered case can be solved using Algorithm 4.7*

If *($k = 1$ and $l = 2$) or ($k = 2$ and $l = 1$) or ($k = l = 1$) or ($k = l = 2$) or ($k, l < 1$)* ***then*** *execute Algorithm 4.7 and return its result.*

`Step 2.` *Simplify the problem*

(i) $a := 1$; $b := 1$;

(ii) ***If*** *($k > 1$)* ***then***

- $\lambda_1 := 1$;
- $a := 2$;

(iii) ***If*** *($l > 1$)* ***then***

- $\mu_1 := 1$;
- $b := 2$.

`Step 3.` *For* $h = 1, 2, \ldots, d$, *and* $i = 1, 2, \ldots, k, m - l, m - l + 1, \ldots, m - 1$, *compute*

(i) $\dfrac{\partial r_{ih}}{\partial \lambda_u} \quad (u = a, a + 1, \ldots, k)$ *by* (4.11)–(4.13);

(ii) $\dfrac{\partial r_{ih}}{\partial \mu_v}$ $(v = b, b+1, \ldots, l)$ *by* (4.14)–(4.16).

`Step 4.` *Compute* ϕ_{ij} $(i = k+1, k+2, \ldots, m-l-1;\ j = 0, 1, \ldots, n)$ *using Algorithm 3.16 and the formula* (3.20).

`Step 5.` *For* $h = 1, 2, \ldots, d$, *and* $i = k+1, k+2, \ldots, m-l-1$, *compute*

(i) $\dfrac{\partial r_{ih}}{\partial \lambda_u}$ $(u = a, a+1, \ldots, k)$ *by* (4.17);

(ii) $\dfrac{\partial r_{ih}}{\partial \mu_v}$ $(v = b, b+1, \ldots, l)$ *by* (4.18).

`Step 6.` *Obtain* $\lambda_a, \lambda_{a+1}, \ldots, \lambda_k$, *and* $\mu_b, \mu_{b+1}, \ldots, \mu_l$ *by solving the linear system*

$$\begin{cases} \displaystyle\sum_{h=1}^{d} \sum_{j=u}^{m-l-1} \left[F_{mj}(\mathbf{r}^h) - F_{nj}(\mathbf{p}^h)\right] \frac{\partial r_{jh}}{\partial \lambda_u} = 0 \qquad (u = a, a+1, \ldots, k), \\ \displaystyle\sum_{h=1}^{d} \sum_{j=k+1}^{m-v} \left[F_{mj}(\mathbf{r}^h) - F_{nj}(\mathbf{p}^h)\right] \frac{\partial r_{jh}}{\partial \mu_v} = 0 \qquad (v = b, b+1, \ldots, l), \end{cases}$$

where $F_{tj}(\mathbf{q})$ *is computed by* (4.9).

`Step 7.` *Check if the solution is feasible*

If *(*$\lambda_1 \le 0$ *or* $\mu_1 \le 0$*)* ***then***

(i) $c := \{\lambda_1 \ge z_0,\ \mu_1 \ge z_1\}$;

(ii) ***If*** *(*$k \ne 1$*)* ***then*** $c := c \setminus \{\lambda_1 \ge z_0\}$;

(iii) ***If*** *(*$l \ne 1$*)* ***then*** $c := c \setminus \{\mu_1 \ge z_1\}$;

(iv) *compute* $E^{(\alpha,\beta)}(\lambda_1, \lambda_2, \ldots, \lambda_k, \mu_1, \mu_2, \ldots, \mu_l)$ *by* (4.6);

(v) *obtain* $\lambda_a, \lambda_{a+1}, \ldots, \lambda_k$, *and* $\mu_b, \mu_{b+1}, \ldots, \mu_l$ *by solving the quadratic programming problem of minimizing the error* $E^{(\alpha,\beta)}$ *subject to the constraints* c.

Phase B

`Step 8.` *Execute Algorithm 4.5 with the computed values of the continuity parameters, and return the solution, i.e., the control points* $r_0, r_1, \ldots, r_m$.

4.4.3 $G^{k,l}$-constrained degree reduction algorithm

Finally, we present Algorithm 4.9 which solves the most general version of the $G^{k,l}$-constrained degree reduction problem (see Problem 1.9), using Algorithms 3.16, 4.5, 4.7 and the method described in §4.2.1. The algorithm works for any k and l not exceeding 3. The computations are organized in two phases which are analogical to the ones mentioned in §4.4.2.

Since Algorithm 4.7 uses explicit formulas, it is executed by Algorithm 4.9 whenever possible (see Algorithm 4.9, Step 1). Obviously, Algorithm 4.9 is, in general, computationally more expensive than the previous algorithms. However, it produces the most accurate results because no simplifying assumptions are made.

Algorithm 4.9. *[$G^{k,l}$-constrained degree reduction of Bézier curves]*

Input: α, β – *parameters of the weighted* L_2*-norm* (1.13)*;*

n, $p_0, p_1, \ldots, p_n$ – *degree and control points of the original Bézier curve* (1.17)*;*

m – *degree of the reduced Bézier curve* (1.18)*;*

k, l – *orders of the geometric continuity (see* (1.20)*);*

z_0, z_1 – *lower bounds for the parameters* λ_1 *and* μ_1*, respectively (see* (4.19)*)*

Assumptions: $n > m > 0$; $z_0, z_1 > 0$; $\alpha, \beta > -1$; $k + l < m - 1$; $-1 \leq k, l < 3$

Output: control points of the $G^{k,l}$*-constrained degree reduced Bézier curve*

Phase A

Step 1. *Check if the considered case can be solved using Algorithm 4.7*

If *(*$k, l < 1$ *or* $k = l = 1$*)* ***then*** *execute Algorithm 4.7 and return its result.*

Step 2. *Compute* ϕ_{ij} *(*$i = k+1, k+2, \ldots, m-l-1$; $j = 0, 1, \ldots, n$*) using Algorithm 3.16 and the formula* (3.20)*.*

Step 3. *Compute* $E^{(\alpha,\beta)}(\lambda_1, \lambda_2, \ldots, \lambda_k, \mu_1, \mu_2, \ldots, \mu_l)$ *by* (4.6)*.*

Step 4. *Determine set* c *of constraints*

(i) $c := \{\lambda_1 \geq z_0,\ \mu_1 \geq z_1\}$;

(ii) ***If*** *(*$k < 1$*)* ***then*** $c := c \setminus \{\lambda_1 \geq z_0\}$;

(iii) ***If*** *(*$l < 1$*)* ***then*** $c := c \setminus \{\mu_1 \geq z_1\}$.

Step 5.

If *(*$k > 1$ *or* $l > 1$*)* ***then***

- *obtain* $\lambda_1, \lambda_2, \ldots, \lambda_k$, *and* $\mu_1, \mu_2, \ldots, \mu_l$ *by solving the nonlinear programming problem of minimizing the error* $E^{(\alpha,\beta)}$ *subject to the constraints* c*;*

else

- *obtain* $\lambda_1, \lambda_2, \ldots, \lambda_k$, *and* $\mu_1, \mu_2, \ldots, \mu_l$ *by solving the quadratic programming problem of minimizing the error* $E^{(\alpha,\beta)}$ *subject to the constraints* c*.*

Phase B

Step 6. *Execute Algorithm 4.5 with the computed values of the continuity parameters, and return the solution, i.e., the control points* $r_0, r_1, \ldots, r_m$*.*

4.5 Examples

This subsection provides of the application of Algorithms 4.7–4.9. For each example, we give weighted L_2-error $E_2^{(\alpha,\beta)}$ (see (4.5)) and maximum error E_∞ (see (1.11)).

In the experiments, we consider the *natural choices* of α and β, i.e.,

$$(\alpha, \beta) \in \left\{(0,0), \left(\frac{1}{2}, \frac{1}{2}\right), \left(-\frac{1}{2}, \frac{1}{2}\right), \left(\frac{1}{2}, -\frac{1}{2}\right), \left(-\frac{1}{2}, -\frac{1}{2}\right)\right\},$$

and set the lower bounds z_0, z_1 of λ_1, μ_1 to 10^{-4} (see (4.19)). Taking into account the different types of continuity constraints, we compare the following cases:

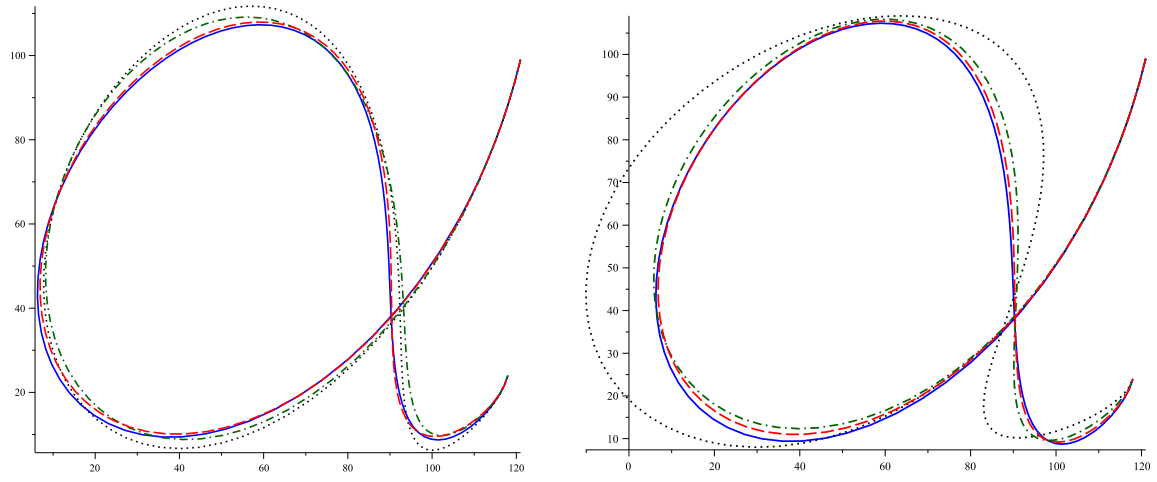
(i) $C^{k,l}$-constrained case, solved using Algorithm 4.7;

(ii) $C^{p,q}/G^{k,l}$-constrained case, solved using Algorithms 4.7 and 4.8;

(iii) $G^{k,l}$-constrained case, solved using Algorithm 4.9.

The results have been obtained on a computer with `Intel Core i5-3337U 1.8GHz` processor and `8GB` of `RAM`, using Maple$^{\text{TM}}$13 with 32-digit arithmetic. We use Maple$^{\text{TM}}$ `fsolve` procedure, in the $C^{p,q}/G^{k,l}$-constrained case, to solve a system of linear equations, and `QPSolve`, `NLPSolve` procedures, to solve quadratic and nonlinear programming problems, respectively. `QPSolve` uses an iterative active set method, and for `NLPSolve` we select `sqp` method (see §4.2.1). Initial points for both procedures correspond to the values of continuity parameters in the $C^{k,l}$-constrained case (see Remark 2.3). Obviously, for the selected cases of continuity constraints (see Remark 4.4), explicit formulas are used, since Algorithms 4.8 and 4.9 execute Algorithm 4.7 whenever possible.

Example 4.10. First, let us revisit Examples 1.5 and 1.6, i.e., consider the Bézier curve "alpha" of degree 11. The results of degree reduction are given in Table 3. Figures 8a and 8b illustrate two of the considered cases. One can see that when it comes to minimizing E_∞ error, usually a good choice is $\alpha = \beta = -\frac{1}{2}$. As expected, the solutions of the $G^{k,l}$-constrained case are the most accurate, while the $C^{p,q}/G^{k,l}$-constrained approach gives less precise results. The $C^{k,l}$ conditions tend to be too restrictive, especially for k or l exceeding 2.



(a) (b)

Figure 8: Degree reduction of 11th degree Bézier curve (blue solid line) to 7th degree Bézier curve with $C^{k,l}$ (black dotted line), $C^{p,q}/G^{k,l}$ (green dash-dotted line) and $G^{k,l}$ (red dashed line) continuity constraints; parameters: (a) $\alpha = \beta = -\frac{1}{2}$, $p = q = 1$, $k = l = 2$, and (b) $\alpha = \beta = -\frac{1}{2}$, $p = 1$, $q = -$, $k = 3$, $l = 1$.

Parameters							$C^{k,l}$ solution		$C^{p,q}/G^{k,l}$ solution		$G^{k,l}$ solution	
m	k	l	p	q	α	β	$E_2^{(\alpha,\beta)}$	E_∞	$E_2^{(\alpha,\beta)}$	E_∞	$E_2^{(\alpha,\beta)}$	E_∞
7	2	2	1	1	0	0	3.73e+0	5.97e+0	2.83e+0	5.27e+0	9.31e−1	2.26e+0
					$-\frac{1}{2}$	$-\frac{1}{2}$	5.75e+0	5.83e+0	4.40e+0	5.14e+0	1.67e+0	1.91e+0
					$-\frac{1}{2}$	$\frac{1}{2}$	3.83e+0	7.53e+0	2.62e+0	6.42e+0	7.85e−1	2.91e+0
					$\frac{1}{2}$	$-\frac{1}{2}$	3.83e+0	7.69e+0	3.18e+0	5.18e+0	1.26e+0	2.19e+0
					$\frac{1}{2}$	$\frac{1}{2}$	2.43e+0	6.10e+0	1.83e+0	5.40e+0	5.32e−1	2.54e+0
7	3	1	1	–	0	0	9.13e+0	1.62e+1	2.51e+0	5.11e+0	1.02e+0	2.41e+0
					$-\frac{1}{2}$	$-\frac{1}{2}$	1.40e+1	1.67e+1	4.07e+0	4.95e+0	1.81e+0	2.07e+0
					$-\frac{1}{2}$	$\frac{1}{2}$	9.41e+0	1.95e+1	2.18e+0	6.38e+0	7.45e−1	3.11e+0
					$\frac{1}{2}$	$-\frac{1}{2}$	9.17e+0	1.88e+1	2.98e+0	4.91e+0	1.45e+0	1.99e+0
					$\frac{1}{2}$	$\frac{1}{2}$	6.00e+0	1.58e+1	1.56e+0	5.25e+0	5.88e−1	2.69e+0

Table 3: Weighted L_2-errors and maximum errors of degree reduction of 11th degree Bézier curve "alpha".

Example 4.11. Now, we apply the algorithms to the Bézier curve "heart" of degree 13 (for the control points, see [92, Appendix B]), and consider the case of $k = l = 2$. The results are presented in Table 4. Notice that the case of $\alpha = \beta = 0$ was also considered in [92, §5.2] and [114, Example 4]. As in [114], we can clearly see that the solution of the $G^{2,2}$-constrained case, in this thesis obtained using Algorithm 4.9, is more accurate than the result given by the approach proposed in [92], which leads to the $C^{1,1}/G^{2,2}$-constrained case (the same as for Algorithms 4.7 and 4.8). As we consider different weight functions, it can be seen that the best choice to minimize E_∞ is $\alpha = \beta = -\frac{1}{2}$. Figure 9 presents the case of $\alpha = \beta = 0$.

Next, we focus on the running times. For the comparison of the $G^{k,l}$-constrained algorithms, see Table 5. Notice that, in some cases, Algorithm 4.9 is slightly faster than the methods from [114]. We use Maple™ `fsolve` procedure to solve the cubic equation [114, (23)] associated with the $G^{2,1}$-constrained case. Implementation of the $G^{2,2}$-constrained method from [114] requires an *unconstrained nonlinear programming solver*. According to the experiments, the *nonlinear simplex method* (`NLPSolve` command with option `method = nonlinearsimplex` and the initial point $\lambda = \eta = 1$) is the fastest solver available in Maple™13. Therefore, we use this solver for the purpose of the comparison. It is worth mentioning that the authors of [114] have omitted the constraints (4.19). Consequently, in some rare cases, the resulting curve may not preserve original tangent directions at the endpoints (see Remark 4.3). To avoid this issue, one should implement the improvements proposed by Lu [74].

Parameters			$C^{2,2}$ solution		$C^{1,1}/G^{2,2}$ solution		$G^{2,2}$ solution	
m	α	β	$E_2^{(\alpha,\beta)}$	E_∞	$E_2^{(\alpha,\beta)}$	E_∞	$E_2^{(\alpha,\beta)}$	E_∞
8	0	0	$1.52e{+}0$	$2.52e{+}0$	$6.36e{-}1$	$1.12e{+}0$	$3.57e{-}1$	$7.14e{-}1$
	$-\frac{1}{2}$	$-\frac{1}{2}$	$2.37e{+}0$	$2.39e{+}0$	$1.05e{+}0$	$1.00e{+}0$	$6.16e{-}1$	$5.28e{-}1$
	$-\frac{1}{2}$	$\frac{1}{2}$	$1.58e{+}0$	$3.34e{+}0$	$6.42e{-}1$	$1.55e{+}0$	$4.23e{-}1$	$9.40e{-}1$
	$\frac{1}{2}$	$-\frac{1}{2}$	$1.52e{+}0$	$3.48e{+}0$	$7.44e{-}1$	$1.31e{+}0$	$3.65e{-}1$	$8.90e{-}1$
	$\frac{1}{2}$	$\frac{1}{2}$	$9.79e{-}1$	$2.64e{+}0$	$3.89e{-}1$	$1.24e{+}0$	$2.11e{-}1$	$9.03e{-}1$

Table 4: Weighted L_2-errors and maximum errors of degree reduction of 13th degree Bézier curve "heart".

Parameters			Running times [ms]	
m	k	l	Algorithm 4.9	Zhou et al. [114]
8	2	1	92	108
10	2	1	121	137
12	2	1	168	166
8	2	2	153	204
10	2	2	298	248
12	2	2	290	292

Table 5: Running times of the $G^{k,l}$-constrained degree reduction of 13th degree Bézier curve "heart"; parameters: $\alpha = \beta = 0$.

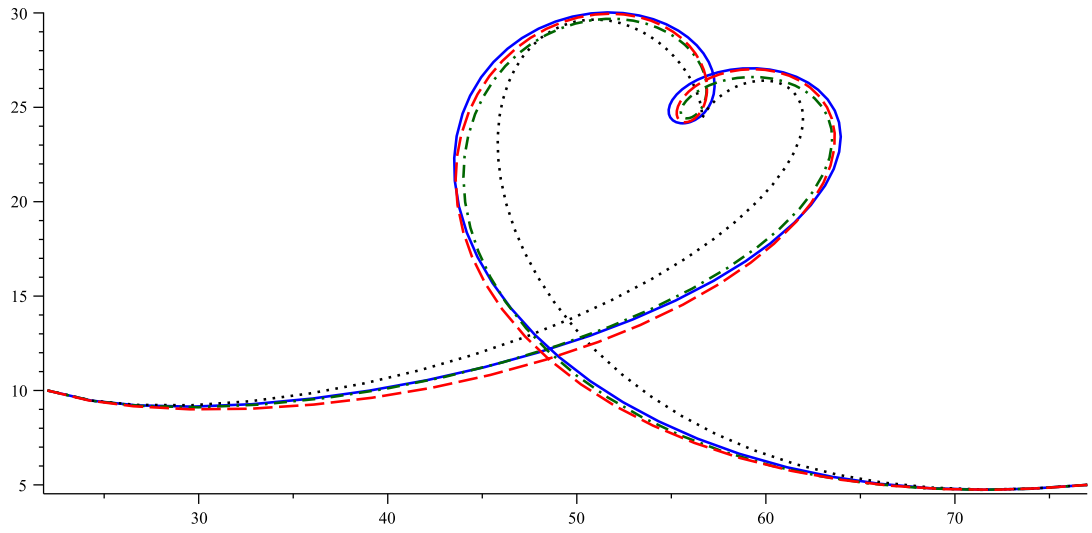


Figure 9: Degree reduction of 13th degree Bézier curve (blue solid line) to 8th degree Bézier curve with $C^{2,2}$ (black dotted line), $C^{1,1}/G^{2,2}$ (green dash-dotted line) and $G^{2,2}$ (red dashed line) continuity constraints; parameters: $\alpha = \beta = 0$.

Example 4.12. Finally, we give the composite Bézier curve "dolphin" obtained by joining nine Bézier curves (for the control points, see `http://www.ii.uni.wroc.pl/~pgo/dolphin.txt`). We apply the algorithms independently to each Bézier curve and compare the quality of the results in Table 6. For each curve, we set $\alpha = \beta := -\frac{1}{2}$, which as we found out, is usually the best choice to minimize E_∞ error. The results confirm the advantage of Algorithm 4.9 over the other ones. The degree reduced composite Bézier curves are shown in Figure 10.

		Parameters						$C^{k,l}$ solution		$C^{p,q}/G^{k,l}$ solution		$G^{k,l}$ solution	
	Curves	n	m	k	l	p	q	$E_2^{(\alpha,\beta)}$	E_∞	$E_2^{(\alpha,\beta)}$	E_∞	$E_2^{(\alpha,\beta)}$	E_∞
Dolphin	Forehead	8	4	2	0	1	–	8.75e+0	8.71e+0	3.65e+0	3.43e+0	3.51e+0	3.52e+0
	Mouth	18	7	1	3	–	1	3.35e+1	4.09e+1	7.86e+0	1.04e+1	6.34e+0	8.08e+0
	Flipper: part 1	12	5	1	2	–	1	6.23e+0	6.84e+0	5.45e+0	5.50e+0	3.60e+0	3.63e+0
	Flipper: part 2	14	5	2	1	1	–	9.94e+0	1.00e+1	7.26e+0	6.83e+0	5.34e+0	5.49e+0
	Tail: part 1	10	5	3	0	1	–	2.66e+1	2.93e+1	2.28e+1	2.25e+1	5.08e+0	5.40e+0
	Tail: part 2	12	5	1	2	–	1	8.35e+0	9.32e+0	4.04e+0	4.28e+0	3.82e+0	3.91e+0
	Back	8	5	2	1	1	–	3.35e+0	4.03e+0	1.67e+0	2.03e+0	1.52e+0	1.56e+0
	Dorsal fin: part 1	6	5	2	0	1	–	4.77e−1	4.73e−1	2.27e−1	1.98e−1	2.25e−1	1.98e−1
	Dorsal fin: part 2	13	6	0	3	–	1	7.54e+0	8.18e+0	2.25e+0	2.47e+0	1.48e+0	1.73e+0

Table 6: Weighted L_2-errors and maximum errors of degree reduction of nine-segment composite Bézier curve "dolphin".

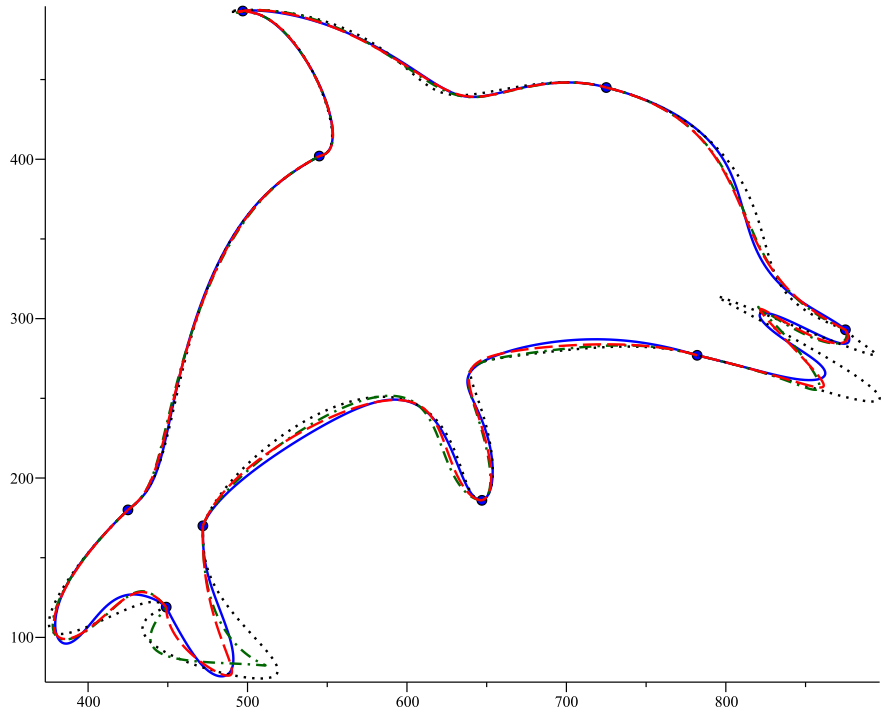


Figure 10: Degree reduction of nine-segment composite Bézier curve (blue solid line) to degree reduced composite Bézier curve with $C^{k,l}$ (black dotted line), $C^{p,q}/G^{k,l}$ (green dash-dotted line) and $G^{k,l}$ (red dashed line) continuity constraints imposed for each segment of the composite curve. The parameters are specified in Table 6.

Chapter 5

Degree reduction of planar Bézier curves with box constraints

In this chapter, we propose a new approach to the problem of $C^{k,l}$-constrained degree reduction of planar Bézier curves with respect to the least squares norm. First, we give illustrative example to show that the solution of the conventional degree reduction problem may not be suitable for further modification and applications (see §5.1). In order to eliminate those issues, we formulate a new problem of degree reduction (see §5.2) and present two methods of solving it. The first one is based on *quadratic programming approach* (see §5.3) and the other one on *BVLS algorithm* [93] (see §5.4). Additionally, we give two methods of solving the so-called *subproblem* which is an essential part of BVLS algorithm (see §5.5). In §5.6, we show more examples to justify the new idea. The results given in this chapter are based on papers [43, 46].

5.1 Motivation

Let us recall that the conventional degree reduction of Bézier curves is to minimize a chosen error function subject to some continuity constraints at the endpoints (see §1.6.1). For example, let us consider the following degree reduction problem (cf. Problem 1.8).

Problem 5.1. *[Traditional degree reduction of planar Bézier curves]*
Given a Bézier curve $P \in \Pi_n^2$,

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t) \qquad (0 \le t \le 1),$$

find a Bézier curve $R \in \Pi_m^2$,

$$R(t) := \sum_{i=0}^{m} r_i B_i^m(t) \qquad (0 \le t \le 1;\ m < n),$$

satisfying the following conditions:

(i) least squares error

$$E \equiv \|P - R\|_2 := \sqrt{\sum_{h=0}^{N} \|P(t_h) - R(t_h)\|^2} \tag{5.1}$$

is minimized, where $\{t_h\}_{h=0}^{N}$ $(N \in \mathbb{N})$ *is a given strictly increasing sequence whose elements are in the interval* $[0, 1]$*;*

(ii) P *and* R *are* $C^{k,l}$*-continuous* ($k, l \geq -1$ *and* $k + l < m - 1$) *at the endpoints, i.e.,*

$$\left.\begin{aligned} P^{(i)}(0) &= R^{(i)}(0) \qquad (i = 0, 1, \ldots, k), \\ P^{(j)}(1) &= R^{(j)}(1) \qquad (j = 0, 1, \ldots, l). \end{aligned}\right\} \tag{5.2}$$

Remark 5.2. A designer that modifies control points uses the convex hull property (see §1.4, pt. 2), which gives an intuition on shape and location of a curve. It can be said that the size of the convex hull is a measure of *predictability* of the curve. As we shall see, the traditional approach can give precise results, however, there is no control over the location of the resulting control points. Consequently, those points may lie far away from the plot of the curve. Such a drawback can significantly disturb further modeling of the curve.

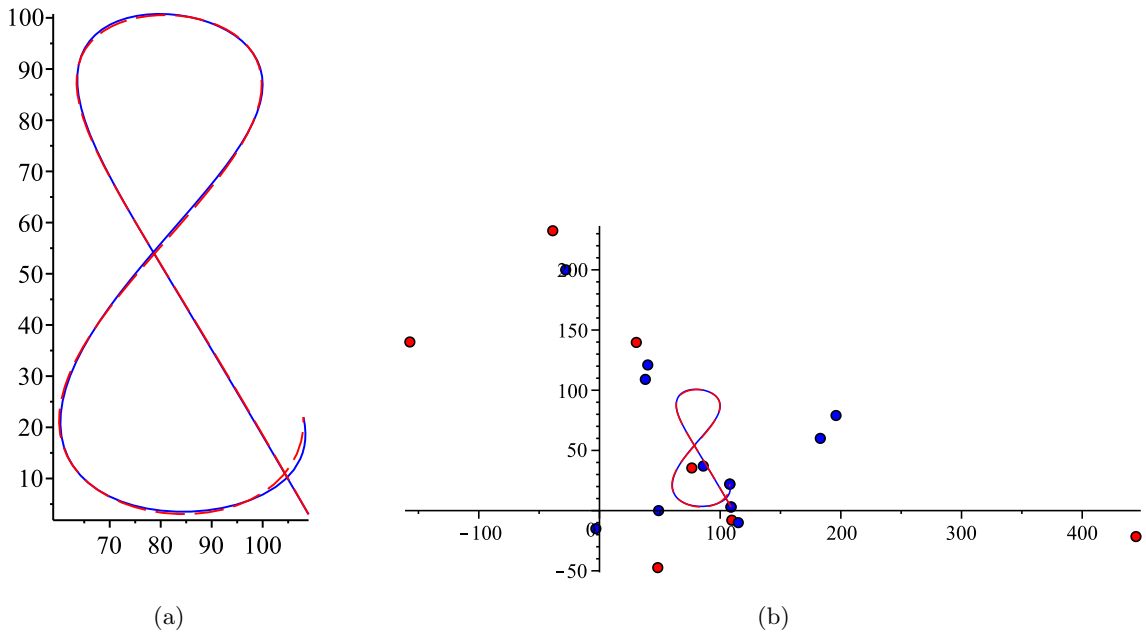
Remark 5.3. Notice that simple tests based on the convex hull property can give quick solutions of problems which usually require more expensive calculation. For example, if convex hulls of two curves do not intersect, which is fairly easy to determine, then the curves do not intersect, which normally is more expensive to decide. *Intersection problem* has many practical applications. Suppose that two Bézier curves represent paths of robotic arms. To avoid collisions, one should solve the intersection problem (see [33, p. 49]). Analogously, one can quickly check that a curve and a surface do not intersect, or decide that a point does not lie on a curve. This strategy can be useful, but only if control points are not far away from a curve and the corresponding convex hull is not too big.

In order to see the issue clearly, let us consider the following example.

Example 5.4. We give the Bézier curve "Ampersand" of degree 10, defined by the control points $(109, 3)$, $(40, 121)$, $(86, 37)$, $(-28, 200)$, $(183, 60)$, $(196, 79)$, $(38, 109)$, $(-3, -15)$, $(49, 0)$, $(115, -10)$, $(108, 22)$. Assuming that $k = l = 0$, $t_h = h/14$ $(h = 0, 1, \ldots, 14)$ (cf. (5.1), (5.2)); we look for a Bézier curve of degree 8 being the solution of Problem 5.1.

Figure 11a shows the plots of both curves. The approximation is *quite precise* (errors: $E = 1.26e{+}0$ and $E_\infty = 1.27e{+}0$; see (1.11)), but this solution has the *defect* mentioned in Remarks 5.2 and 5.3. See Figure 11b which presents the location of the control points of both curves. Clearly, the resulting ones are located far away from the plot of the degree reduced curve. Furthermore, the convex hull of the resulting curve is much bigger than the one of the original curve, and useless in finding quick solutions of the problems mentioned earlier (see Remark 5.3).

Now, let us impose some additional constraints on the searched control points. We enforce their location inside the specified rectangular area (including edges of the rectangle). Further on in this thesis, such restrictions are called box constraints. This time, we obtain the degree reduced curve having more *intuitive* location of the control points (see Figure 12). As a result, it can be easily modified, which is desired from a practical point of view. In addition, this approach leads to much smaller convex hull. Obviously, the errors must be inevitably larger than for the traditional degree reduction because we have imposed the additional constraints (errors: $E = 4.18e{+}0$ and $E_\infty = 4.16e{+}0$). However, the result is still satisfying.



(a) (b)

Figure 11: The original Bézier curve (blue solid line with blue control points) and the degree reduced Bézier curve (red dashed line with red control points) being the solution of Problem 5.1.

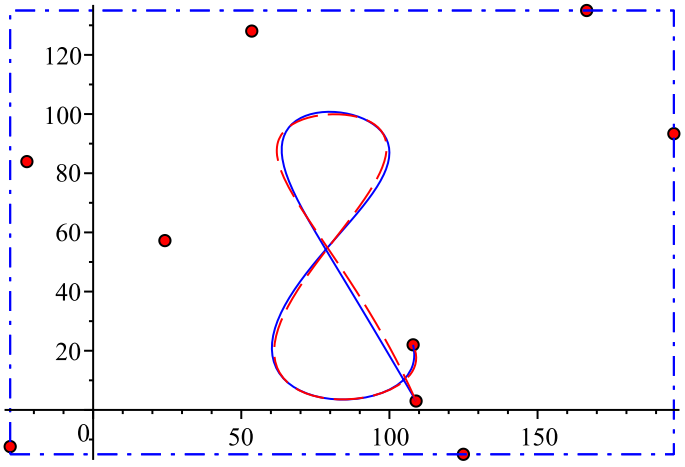


Figure 12: The original Bézier curve (blue solid line) and the degree reduced Bézier curve (red dashed line with red control points) satisfying the conditions (i), (ii) and box constraints (blue dotted-dashed frame).

5.2 Problem of degree reduction of planar Bézier curves with box constraints

Taking into account the remarks presented in the previous subsection, we formulate the following problem of degree reduction of planar Bézier curves (cf. Problem 5.1).

Problem 5.5. *[Degree reduction of planar Bézier curves with box constraints]*
Let there be given a Bézier curve $P = [P_x, P_y] \in \Pi_n^2$,

$$P(t) := \sum_{i=0}^{n} p_i B_i^n(t) \qquad (0 \le t \le 1),$$

where $p_i := (p_i^x, p_i^y) \in \mathbb{R}^2$. *Find a Bézier curve* $R = [R_x, R_y] \in \Pi_m^2$,

$$R(t) := \sum_{i=0}^{m} r_i B_i^m(t) \qquad (0 \le t \le 1;\ m < n),$$

satisfying the following conditions:

(i) least squares error

$$E \equiv \|P - R\|_2 = \sqrt{\sum_{h=0}^{N} \|P(t_h) - R(t_h)\|^2} \tag{5.3}$$

is minimized, where $\{t_h\}_{h=0}^{N}$ $(N \in \mathbb{N})$ *is a given strictly increasing sequence whose elements are in the interval* $[0, 1]$*;*

(ii) P *and* R *are* $C^{k,l}$*-continuous* ($k, l \ge -1$ *and* $k + l < m - 1$) *at the endpoints, i.e.,*

$$\left.\begin{aligned} P^{(i)}(0) &= R^{(i)}(0) \qquad (i = 0, 1, \ldots, k), \\ P^{(j)}(1) &= R^{(j)}(1) \qquad (j = 0, 1, \ldots, l); \end{aligned}\right\} \tag{5.4}$$

(iii) control points $r_i := (r_i^x, r_i^y)$ $(k < i < m - l)$ *are located inside the specified rectangular area, including edges of the rectangle, i.e., the following box constraints are fulfilled:*

$$c_z \le r_i^z \le C_z \qquad (i = k + 1, k + 2, \ldots, m - l - 1;\ z = x, y), \tag{5.5}$$

where $c_x, c_y, C_x, C_y \in \mathbb{R}$.

To begin with, recall that the continuity conditions (5.4) imply the well-known formulas (2.3) and (2.4) for the control points $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$, respectively. For details, see §2.1.

Next, it is easy to check that

$$E^2 = E_x^2 + E_y^2,$$

where

$$E_z := \sqrt{\sum_{h=0}^{N} \left(P_z(t_h) - R_z(t_h)\right)^2} \qquad (z = x, y).$$

Consequently, the solution can be obtained in a *componentwise way*. Therefore, it is sufficient to explain how to compute $r_{k+1}^x, r_{k+2}^x, \ldots, r_{m-l-1}^x$ in order to minimize E_x with the box constraints for $z = x$ (cf. (5.5)).

5.3 Degree reduction using quadratic programming approach

In this subsection, we use the quadratic programming approach to solve Problem 5.5.

Quadratic programming is an *optimization problem* of minimizing (or maximizing) a quadratic *objective function* f of several variables $\mathbf{x} \in \mathbb{R}^i$ subject to linear constraints on these variables. More precisely, we look for the extremum of function

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}, \tag{5.6}$$

where $\mathbf{Q} \in \mathbb{R}^{i\times i}$, $\mathbf{c} \in \mathbb{R}^i$, subject to the following constraints:

$$\mathbf{G}\mathbf{x} \leq \mathbf{h}, \qquad \mathbf{x} \geq \mathbf{0}, \tag{5.7}$$

where $\mathbf{G} \in \mathbb{R}^{j\times i}$ and $\mathbf{h} \in \mathbb{R}^j$. Here the notation $\mathbf{v} \leq \mathbf{w}$ means that $v_i \leq w_i$ $(i = 1, 2, \ldots, q)$, where $\mathbf{v} := [v_1, v_2, \ldots, v_q]^T \in \mathbb{R}^q$ and $\mathbf{w} := [w_1, w_2, \ldots, w_q]^T \in \mathbb{R}^q$. The definition of $\mathbf{v} \geq \mathbf{w}$ is similar.

Before we proceed further, we provide some other useful definitions and notation. We define vectors of coordinates of the original and searched control points,

$$\mathbf{p}_x := [p_0^x, p_1^x, \ldots, p_n^x]^T, \qquad \mathbf{r}_x := [r_0^x, r_1^x, \ldots, r_m^x]^T,$$

respectively; vectors of lower and upper bounds (cf. (5.5)),

$$\mathbf{l}_x := [c_x, c_x, \ldots, c_x]^T \in \mathbb{R}^{m-l-k-1}, \qquad \mathbf{u}_x := [C_x, C_x, \ldots, C_x]^T \in \mathbb{R}^{m-l-k-1},$$

respectively; and a vector of Bernstein polynomials of degree n evaluated at a point t,

$$\mathbf{b}_{n,t} := [B_0^n(t), B_1^n(t), \ldots, B_n^n(t)].$$

In order to obtain selected *submatrices* and *subvectors*, we will use the following notation. Let $\mathbf{M} \in \mathbb{R}^{n\times m}$ be a matrix, and let $\mathcal{A} := \{i_1, i_2, \ldots, i_\alpha\} \subset [0, n-1]$, $\mathcal{B} := \{j_1, j_2, \ldots, j_\beta\} \subset [0, m-1]$ be sets of natural numbers sorted in ascending order. The notation

$$\mathbf{M}^{\mathcal{A},\mathcal{B}} \tag{5.8}$$

defines a matrix formed by rows $i_1+1, i_2+1, \ldots, i_\alpha+1$ and columns $j_1+1, j_2+1, \ldots, j_\beta+1$ of the matrix $\mathbf{M}$. Similarly, we use $\mathbf{v}^{\mathcal{A}}$, where $\mathbf{v}$ is a vector in $\mathbb{R}^n$.

Now, a goal is to represent the *significant terms* of $E_x^2 \equiv E_x^2(\mathbf{r}_x)$ in the form (5.6). First, notice that

$$\begin{aligned}
E_x^2(\mathbf{r}_x) &= \sum_{h=0}^{N} \left(\mathbf{b}_{n,t_h}\mathbf{p}_x - \mathbf{b}_{m,t_h}\mathbf{r}_x\right)^2 = \sum_{h=0}^{N} \left(\mathbf{b}_{n,t_h}\mathbf{p}_x - \mathbf{b}^{\mathcal{D}}_{m,t_h}\mathbf{r}_x^{\mathcal{D}} - \mathbf{b}^{\mathcal{C}}_{m,t_h}\mathbf{r}_x^{\mathcal{C}}\right)^2 \\
&= \sum_{h=0}^{N} \left(\mathbf{b}^{\mathcal{D}}_{m,t_h}\mathbf{r}_x^{\mathcal{D}}\right)^2 - 2\sum_{h=0}^{N} \left(\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)\left(\mathbf{b}^{\mathcal{D}}_{m,t_h}\mathbf{r}_x^{\mathcal{D}}\right) - \left(\mathbf{b}^{\mathcal{C}}_{m,t_h}\mathbf{r}_x^{\mathcal{C}}\right)\left(\mathbf{b}^{\mathcal{D}}_{m,t_h}\mathbf{r}_x^{\mathcal{D}}\right)\right) \\
&\quad + \sum_{h=0}^{N} \left(\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)^2 + \left(\mathbf{b}^{\mathcal{C}}_{m,t_h}\mathbf{r}_x^{\mathcal{C}}\right)^2 - 2\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)\left(\mathbf{b}^{\mathcal{C}}_{m,t_h}\mathbf{r}_x^{\mathcal{C}}\right)\right) \\
&=: g\left(\mathbf{r}_x^{\mathcal{D}}\right) + \sum_{h=0}^{N} \left(\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)^2 + \left(\mathbf{b}^{\mathcal{C}}_{m,t_h}\mathbf{r}_x^{\mathcal{C}}\right)^2 - 2\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)\left(\mathbf{b}^{\mathcal{C}}_{m,t_h}\mathbf{r}_x^{\mathcal{C}}\right)\right),
\end{aligned} \tag{5.9}$$

where $\mathcal{C} := \{0, 1, \ldots, k, m-l, m-l+1, \ldots, m\}$ and $\mathcal{D} := \{k+1, k+2, \ldots, m-l-1\}$. Observe that the term (5.9) is constant, therefore, it is sufficient to minimize g. It is easy to check that

$$g\left(\mathbf{r}_x^{\mathcal{D}}\right) = \mathbf{d}^T \mathbf{r}_x^{\mathcal{D}} + \frac{1}{2}\left(\mathbf{r}_x^{\mathcal{D}}\right)^T \mathbf{Q} \mathbf{r}_x^{\mathcal{D}},$$

where $\mathbf{Q} := [Q_{i,j}] \in \mathbb{R}^{(m-l-k-1)\times(m-l-k-1)}$ and $\mathbf{d} := [d_i] \in \mathbb{R}^{m-l-k-1}$ with

$$Q_{i,j} := 2\sum_{h=0}^{N} B_{i+k}^m(t_h) B_{j+k}^m(t_h) \qquad (i,j = 1, 2, \ldots, m-l-k-1), \tag{5.10}$$

$$d_i := 2\sum_{h=0}^{N} \left(\mathbf{b}_{m,t_h}^{\mathcal{C}} \mathbf{r}_x^{\mathcal{C}} - \mathbf{b}_{n,t_h} \mathbf{p}_x\right) B_{i+k}^m(t_h) \qquad (i = 1, 2, \ldots, m-l-k-1).$$

Note that the constraints (5.5), for $z = x$, are not written in the form (5.7). However, they can be easily adjusted. Assume that $\mathbf{s}_x := \mathbf{r}_x^{\mathcal{D}} - \mathbf{l}_x$ and $\mathbf{h} := \mathbf{u}_x - \mathbf{l}_x$. Then, the problem is to minimize

$$g\left(\mathbf{s}_x + \mathbf{l}_x\right) = \mathbf{c}^T \mathbf{s}_x + \frac{1}{2}\mathbf{s}_x^T \mathbf{Q} \mathbf{s}_x + \mathbf{d}^T \mathbf{l}_x + \frac{1}{2}\mathbf{l}_x^T \mathbf{Q} \mathbf{l}_x$$

with respect to $\mathbf{s}_x$, where $\mathbf{c} := \mathbf{d} + \mathbf{Q}\mathbf{l}_x$, subject to the constraints

$$\mathbf{G}\mathbf{s}_x \le \mathbf{h}, \qquad \mathbf{s}_x \ge \mathbf{0}, \tag{5.11}$$

where $\mathbf{G}$ is the *identity matrix* of size $m-l-k-1$.

Finally, taking into account that the term $\mathbf{d}^T \mathbf{l}_x + \frac{1}{2}\mathbf{l}_x^T \mathbf{Q} \mathbf{l}_x$ is constant, a goal is to find the minimum of

$$f\left(\mathbf{s}_x\right) := \mathbf{c}^T \mathbf{s}_x + \frac{1}{2}\mathbf{s}_x^T \mathbf{Q} \mathbf{s}_x \tag{5.12}$$

subject to the constraints (5.11).

Obviously, if f has a minimum point at $\mathbf{s}_x$ subject to the constraints (5.11), then we can minimize E_x^2 subject to the constraints (5.4) and (5.5) by setting

$$\mathbf{r}_x := \Big[r_0^x, r_1^x, \ldots, r_k^x, \underbrace{r_{k+1}^x, \ldots, r_{m-l-1}^x}_{\mathbf{r}_x^{\mathcal{D}}}, r_{m-l}^x, \ldots, r_m^x\Big]^T,$$

where $\mathbf{r}_x^{\mathcal{D}} \equiv \left[r_{k+1}^x, r_{k+2}^x, \ldots, r_{m-l-1}^x\right]^T := \mathbf{s}_x + \mathbf{l}_x$ (see (2.3) and (2.4)).

It is well known that if matrix $\mathbf{Q}$ is *positive semi-definite* and the *feasible set* (5.7) is nonempty, closed and bounded, then the quadratic programming problem has a solution (see, e.g., [29, §2.3]).

Theorem 5.6. *Matrix* $\mathbf{Q}$ *given by* (5.10) *is positive semi-definite.*

Proof. Let $\mathbf{x} := [x_0, x_1, \ldots, x_{m-l-k-2}]^T$ be any non-zero vector in $\mathbb{R}^{m-l-k-1}$. Notice that

$$\begin{aligned}\mathbf{x}^T \mathbf{Q} \mathbf{x} &= 2\sum_{i=k+1}^{m-l-1} \sum_{j=k+1}^{m-l-1} x_{i-k-1} x_{j-k-1} \sum_{h=0}^{N} B_i^m(t_h) B_j^m(t_h) \\ &= 2\sum_{h=0}^{N} \left(\sum_{i=k+1}^{m-l-1} B_i^m(t_h) x_{i-k-1}\right)^2 \ge 0,\end{aligned}$$

which completes the proof. □

The problem of minimizing (5.12) subject to (5.11) can be solved using, e.g., *Wolfe's method* [99] which is a *simplex-type method* for the quadratic programming. A disadvantage of this method is that it has exponential worst-case complexity. However, it works relatively fast, particularly for small-sized problems. In addition, there are some algorithms solving quadratic programming problems in polynomial time (see [56, 105]), but in the context of the degree reduction problem, their significance is only theoretical. Furthermore, there are some papers dealing with the *quadratic programming problem with box constraints*,

$$\min_{\mathbf{v}\leq\mathbf{x}\leq\mathbf{w}} \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x},$$

where $\mathbf{x}, \mathbf{v}, \mathbf{w}, \mathbf{c} \in \mathbb{R}^i$ and $\mathbf{Q} \in \mathbb{R}^{i\times i}$ (cf. (5.6), (5.7)). To solve such a problem, one can use a variety of strategies, including *active set methods* (see, e.g., [39]) and *interior point algorithms* (see, e.g., [49]). Some of the approaches combine the active set strategy with *gradient projection method* (see, e.g., [80]). For extensive lists of references, see the mentioned papers.

5.4 Degree reduction using BVLS algorithm

In this subsection, we solve Problem 5.5 using BVLS algorithm [93].

BVLS algorithm, which is a generalization of iterative *NNLS algorithm (non-negative least-squares)* [59], solves the so-called *bvls problem (bounded-variable least-squares)* written in the following form:

$$\min_{\mathbf{v}\leq\mathbf{x}\leq\mathbf{w}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|, \tag{5.13}$$

where $\mathbf{A} \in \mathbb{R}^{j\times i}$, $\mathbf{x}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^i$, $\mathbf{b} \in \mathbb{R}^j$. Here $\|\cdot\|$ is the Euclidean vector norm in $\mathbb{R}^j$. The strategy of the algorithm is to find the so-called *active set* for an optimal solution. We call a constraint *active* if it forces a boundary value of a variable. A solution improves in each iteration of the algorithm, until the optimal one is found, which happens in finite number of steps (see [93, §2]).

In our case, the adjustment of Problem 5.5 to the form (5.13) is simple. We assume that $\mathbf{v} := \mathbf{l}_x$, $\mathbf{w} := \mathbf{u}_x$, $\mathbf{x} := \mathbf{r}_x^{\mathcal{D}}$, and $\mathbf{A} := [A_{i,j}] \in \mathbb{R}^{(N+1)\times(m-l-k-1)}$, $\mathbf{b} := [b_i] \in \mathbb{R}^{N+1}$, where

$$\begin{aligned} A_{i,j} &:= B^m_{j+k}(t_{i-1}) \qquad (i = 1, 2, \ldots, N+1;\ j = 1, 2, \ldots, m-l-k-1),\\ b_i &:= \mathbf{b}_{n,t_{i-1}}\mathbf{p}_x - \mathbf{b}^{\mathcal{C}}_{m,t_{i-1}}\mathbf{r}^{\mathcal{C}}_x \qquad (i = 1, 2, \ldots, N+1). \end{aligned}$$

Here we use the notation of (5.8).

We will now describe, step by step, how to solve Problem 5.5 using BVLS algorithm. Note that a more detailed description of the original algorithm, including all necessary formulas, is given in [93].

Step 1. Let us assume that in each iteration of the algorithm, vector $\mathbf{x}$ contains current values of all variables, set $\mathcal{F}$ contains indices of variables satisfying strict version of inequalities (5.5), whereas sets $\mathcal{L}$ and $\mathcal{U}$ contain indices of variables which have reached minimum and maximum permissible value, respectively (cf. (5.5)). At the beginning, we set

$$\mathcal{L}\cup\mathcal{U} := \emptyset, \qquad \mathcal{F} := \{k+1, k+2, \ldots, m-l-1\}, \qquad \mathbf{x} := \frac{\mathbf{l}_x + \mathbf{u}_x}{2},$$

and with such assumptions, in the first iteration, we omit the next step of the original algorithm and go to Step 3.

Step 2. We omit this step in the first iteration of the algorithm. Let

$$\mathbf{g} \equiv [g_{k+1}, g_{k+2}, \ldots, g_{m-l-1}]^T := -\nabla h(\mathbf{x}) \equiv \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}),$$

where $h(\mathbf{x}) := \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$. The algorithm selects a variable from set $\mathcal{L} \cup \mathcal{U}$. The chosen variable is the one that violates the following *Karush-Kuhn-Tucker (KKT) optimality conditions for the bvls problem*:

$$g_i \leq 0 \quad (i \in \mathcal{L}), \qquad g_j \geq 0 \quad (j \in \mathcal{U}), \tag{5.14}$$

the most in terms of the absolute value (for details, see [93, §2, pt. 4]). Next, an index of the selected variable is transferred to set $\mathcal{F}$. If none of the variables violate the above conditions, then $\mathbf{x}$ is the optimal solution of the bvls problem, and the algorithm stops.

Step 3. Next, we solve the subproblem, i.e., we look for the optimal values of variables whose indices are in set $\mathcal{F}$ subject to the fixed values of variables whose indices are in set $\mathcal{L} \cup \mathcal{U}$. More precisely, the subproblem is to minimize the function E_x^2 subject to the constraints (5.4) and the conditions

$$r_i^x = c_x \quad (i \in \mathcal{L}), \qquad r_j^x = C_x \quad (j \in \mathcal{U}) \tag{5.15}$$

(cf. (5.5)). Notice that we ignore the constraints (5.5). Taking into account the arrangements from Step 1, we observe that in the first iteration of the algorithm, the subproblem is equivalent to the traditional degree reduction problem (see Problem 5.1) in the case of $z = x$. In the next iterations, we are dealing with degree reduction problem subject to the constraints (5.4) and (5.15) for $z = x$. See §5.5, where we describe in detail the methods of solving the subproblem.

Step 4. If the obtained values of all variables, whose indices are in set $\mathcal{F}$, satisfy strict version of inequalities (5.5), then one should assign them to $\mathbf{x}$, and go to Step 2 to check the optimality of such a result. Otherwise, we have to *repair* the solution (see Step 5).

Step 5. Now, we repair the solution, i.e., we modify values of variables whose indices are in set $\mathcal{F}$ (for details, see [93, §2, pts. 8–10]). The repair guarantees that *(i)* a variable which was the farthest from fulfilling the constraints now has the boundary value; *(ii)* other variables, whose indices are in set $\mathcal{F}$, satisfy the constraints (5.5). As a result of the repair, an index of at least one variable is transferred from set $\mathcal{F}$ to set $\mathcal{L}$ or $\mathcal{U}$. After determining elements of sets $\mathcal{F}$, $\mathcal{L}$ and $\mathcal{U}$, one should assign new values to $\mathbf{x}$, and go back to Step 3, where the subproblem is solved again.

Remark 5.7. It is worth noting that an approximate solution, sufficiently accurate or received after a specified number of iterations, often only slightly differs in quality from the optimal one. Therefore, a reasonable stopping criterion, alternative to the optimality conditions (5.14), may allow for earlier termination of the algorithm and result in a solution close enough to the optimum.

Since BVLS algorithm is more adjusted to the considered problem, it seems to be a better choice than the previously mentioned approach (see §5.3). The main computational cost of a single iteration of the algorithm is associated with solving the subproblem (see Step 3). In the next subsection, we will focus on finding efficient methods of solving the subproblem.

Finally, we give a block diagram of BVLS algorithm, including the use of alternative stopping criterion (see Figure 13).

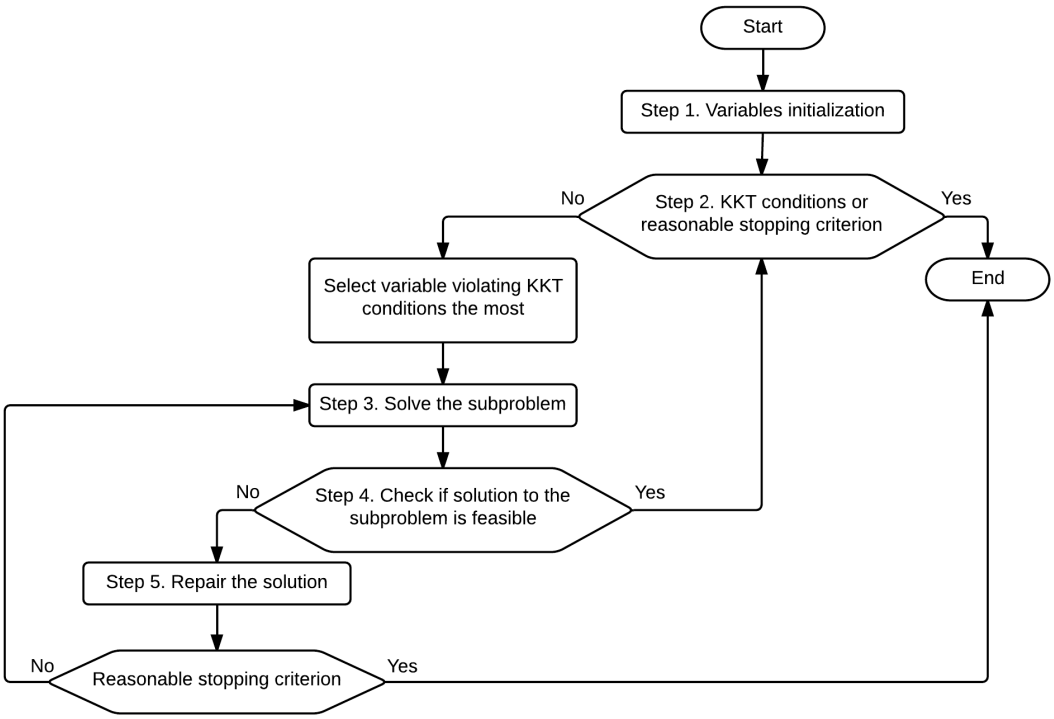


Figure 13: Block diagram of BVLS algorithm.

5.5 Solving the subproblem

In the previous subsection, we describe how to solve Problem 5.5 using BVLS algorithm. An essential part of this algorithm is the subproblem, which is solved in each iteration (see §5.4, Step 3). Let us recall that the subproblem is to minimize E_x^2 subject to the constraints (5.4) and (5.15) for $z = x$. Now, we give two methods of dealing with the subproblem. The first one, proposed in [43, Appendix], solves a system of normal equations, while the other one, presented in [46, §4], is based on the properties of dual bases (see §3.2.3 and §3.2.4).

5.5.1 A straightforward method

Here we give formulas for the solution of the subproblem. The method is based on solving a system of normal equations.

First, we define $\mathcal{H} := \{0, 1, \ldots, n\}$, $\mathcal{K} := \mathcal{L} \cup \mathcal{U} \cup \mathcal{C}$, and $\mathbf{Q}_{n,m} := \left[Q_{i,j}^{n,m}\right] \in \mathbb{R}^{(n+1)\times(m+1)}$, where

$$Q_{i,j}^{n,m} := \sum_{h=0}^{N} B_{i-1}^{n}(t_h) B_{j-1}^{m}(t_h) \qquad (i = 1, 2, \ldots, n+1;\ j = 1, 2, \ldots, m+1).$$

Further on in this subsection, we use the notation of (5.8) to denote matrices formed by specified rows and columns. Analogous notation is used to define vectors.

Now, we write

$$\begin{aligned} E_x^2(\mathbf{r}_x) &= \sum_{h=0}^{N}(\mathbf{b}_{n,t_h}\mathbf{p}_x - \mathbf{b}_{m,t_h}\mathbf{r}_x)^2 = \sum_{h=0}^{N}\left(\mathbf{b}_{n,t_h}\mathbf{p}_x - \mathbf{b}_{m,t_h}^{\mathcal{F}}\mathbf{r}_x^{\mathcal{F}} - \mathbf{b}_{m,t_h}^{\mathcal{K}}\mathbf{r}_x^{\mathcal{K}}\right)^2 \\ &= \sum_{h=0}^{N}\left(\mathbf{b}_{m,t_h}^{\mathcal{F}}\mathbf{r}_x^{\mathcal{F}}\right)^2 - 2\sum_{h=0}^{N}\left(\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)\left(\mathbf{b}_{m,t_h}^{\mathcal{F}}\mathbf{r}_x^{\mathcal{F}}\right) - \left(\mathbf{b}_{m,t_h}^{\mathcal{K}}\mathbf{r}_x^{\mathcal{K}}\right)\left(\mathbf{b}_{m,t_h}^{\mathcal{F}}\mathbf{r}_x^{\mathcal{F}}\right)\right) \\ &\quad + \sum_{h=0}^{N}\left(\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)^2 + \left(\mathbf{b}_{m,t_h}^{\mathcal{K}}\mathbf{r}_x^{\mathcal{K}}\right)^2 - 2\left(\mathbf{b}_{n,t_h}\mathbf{p}_x\right)\left(\mathbf{b}_{m,t_h}^{\mathcal{K}}\mathbf{r}_x^{\mathcal{K}}\right)\right). \end{aligned} \tag{5.16}$$

Next, we omit the constant term (5.16), and solve the subproblem by minimizing

$$\varepsilon\left(\mathbf{r}_x^{\mathcal{F}}\right) := \left(\mathbf{r}_x^{\mathcal{F}}\right)^T \mathbf{Q}_{m,m}^{\mathcal{F},\mathcal{F}}\mathbf{r}_x^{\mathcal{F}} - 2\mathbf{p}_x^T\mathbf{Q}_{n,m}^{\mathcal{H},\mathcal{F}}\mathbf{r}_x^{\mathcal{F}} + 2\left(\mathbf{r}_x^{\mathcal{K}}\right)^T \mathbf{Q}_{m,m}^{\mathcal{K},\mathcal{F}}\mathbf{r}_x^{\mathcal{F}}.$$

In order to find the minimum of ε, it is sufficient to solve the following system of linear equations:

$$\mathbf{0} = \frac{\partial\varepsilon\left(\mathbf{r}_x^{\mathcal{F}}\right)}{\partial\mathbf{r}_x^{\mathcal{F}}}.$$

After calculating the partial derivatives, we obtain the system

$$\mathbf{Q}_{m,m}^{\mathcal{F},\mathcal{F}}\mathbf{r}_x^{\mathcal{F}} = \left(\mathbf{Q}_{n,m}^{\mathcal{H},\mathcal{F}}\right)^T \mathbf{p}_x - \left(\mathbf{Q}_{m,m}^{\mathcal{K},\mathcal{F}}\right)^T \mathbf{r}_x^{\mathcal{K}}, \tag{5.17}$$

and the solution is given by

$$\mathbf{r}_x^{\mathcal{F}} = \left(\mathbf{Q}_{m,m}^{\mathcal{F},\mathcal{F}}\right)^{-1}\left(\left(\mathbf{Q}_{n,m}^{\mathcal{H},\mathcal{F}}\right)^T \mathbf{p}_x - \left(\mathbf{Q}_{m,m}^{\mathcal{K},\mathcal{F}}\right)^T \mathbf{r}_x^{\mathcal{K}}\right). \tag{5.18}$$

Notice that the computation of $\mathbf{r}_x^{\mathcal{F}}$, by the formula (5.18), requires matrix inversion. However, this can be avoided by solving the system (5.17) using, e.g., *LUP decomposition*, which has better numerical properties. In addition, before the first iteration of BVLS algorithm, it is recommended to determine the following matrix and vector:

$$\mathbf{Q}_{m,m}, \quad \left(\mathbf{Q}_{n,m}^{\mathcal{H},\mathcal{D}}\right)^T \mathbf{p}_x,$$

from which – by selecting the appropriate rows or columns – it is possible to obtain matrices and vectors

$$\mathbf{Q}_{m,m}^{\mathcal{F},\mathcal{F}}, \quad \mathbf{Q}_{m,m}^{\mathcal{K},\mathcal{F}}, \quad \left(\mathbf{Q}_{n,m}^{\mathcal{H},\mathcal{F}}\right)^T \mathbf{p}_x,$$

required for the next iterations, without repeating some redundant calculations.

5.5.2 A method based on the properties of dual bases

In §5.5.1, we did not use the fact that consecutive subproblems are *quite similar*. Now, we describe the connection between the subproblems and solve them more efficiently than before.

Let $\mathcal{F}_i$, $\mathcal{L}_i$ and $\mathcal{U}_i$ denote sets $\mathcal{F}$, $\mathcal{L}$ and $\mathcal{U}$, respectively, before solving the subproblem in ith iteration (cf. §5.4, Step 1). The ith subproblem can be formulated in the following way.

Problem 5.8. *[ith subproblem]*
Find the optimal values of variables (i.e., coordinates of the control points) whose indices are in set $\mathcal{F}_i$ subject to the fixed values of variables whose indices are in set $\mathcal{C} \cup \mathcal{L}_i \cup \mathcal{U}_i$. More precisely, we look for $\psi_i^* \in \Pi_m^{\mathcal{F}_i}$ such that:

$$\|\varphi_i - \psi_i^*\|_2 = \min_{\psi_i \in \Pi_m^{\mathcal{F}_i}} \|\varphi_i - \psi_i\|_2, \tag{5.19}$$

where

$$\Pi_m^{\mathcal{A}} := \operatorname{span}\left\{B_j^m \,:\, j \in \mathcal{A}\right\},$$

$$\varphi_i := \sum_{h=0}^{n} p_h^x B_h^n - \sum_{j \in \mathcal{C}} r_j^x B_j^m - \sum_{j \in \mathcal{L}_i} c_x B_j^m - \sum_{j \in \mathcal{U}_i} C_x B_j^m.$$

Obviously, the above formulation is equivalent to the one given in §5.4, Step 3.

Recall that

$$\mathcal{L}_1 = \mathcal{U}_1 := \emptyset, \qquad \mathcal{F}_1 := \{k+1, k+2, \ldots, m-l-1\}$$

(see §5.4, Step 1). Therefore, in the first iteration, we have

$$\varphi_1 := \sum_{h=0}^{n} p_h^x B_h^n - \sum_{j \in \mathcal{C}} r_j^x B_j^m.$$

To solve the first subproblem, we start with the construction of the dual basis for the basis

$$\left\{B_{k+1}^m, B_{k+2}^m, \ldots, B_{m-l-1}^m\right\}$$

of the space $\Pi_m^{\mathcal{F}_1}$, using Algorithm 3.7. Then, we use Fact 3.2 to obtain the best least squares approximation $\psi_1^* \in \Pi_m^{\mathcal{F}_1}$ of φ_1.

Now, let us consider the ith subproblem ($i > 1$) and its relation with the previous one. There are two possibilities.

Case 1. One element q was transferred from $\mathcal{L}_{i-1}$ or $\mathcal{U}_{i-1}$ to $\mathcal{F}_{i-1}$ (see §5.4, Step 2). Therefore, we set $\mathcal{F}_i := \mathcal{F}_{i-1} \cup \{q\}$ and $((\mathcal{L}_i := \mathcal{L}_{i-1} \setminus \{q\} \wedge \mathcal{U}_i := \mathcal{U}_{i-1})$ or $(\mathcal{U}_i := \mathcal{U}_{i-1} \setminus \{q\} \wedge \mathcal{L}_i := \mathcal{L}_{i-1}))$. According to (5.19), for the given $\varphi_i := \varphi_{i-1} + sB_q^m$ ($s = c_x$ or $s = C_x$), we look for the optimal element $\psi_i^* \in \Pi_m^{\mathcal{F}_i}$. Obviously, we have

$$\left\{B_j^m \,:\, j \in \mathcal{F}_i\right\} = \{B_h^m \,:\, h \in \mathcal{F}_{i-1}\} \cup \left\{B_q^m\right\}.$$

Taking into account that in the previous iteration the dual basis for the basis $\{B_h^m : h \in \mathcal{F}_{i-1}\}$ of the space $\Pi_m^{\mathcal{F}_{i-1}}$ was computed, we use Algorithm 3.6 to obtain the dual basis for the basis $\{B_j^m \,:\, j \in \mathcal{F}_i\}$ of the space $\Pi_m^{\mathcal{F}_i}$. Then, we compute the optimal element using Fact 3.2.

Case 2. At least one element was transferred from $\mathcal{F}_{i-1}$ to $\mathcal{L}_{i-1}$ or $\mathcal{U}_{i-1}$ (see §5.4, Step 5). Apart from very rare cases, exactly one element q was transferred. If the rare case occurred, then the procedure given below should be applied repeatedly. We set $\mathcal{F}_i := \mathcal{F}_{i-1} \setminus \{q\}$ and $((\mathcal{L}_i := \mathcal{L}_{i-1} \cup \{q\} \wedge \mathcal{U}_i := \mathcal{U}_{i-1})$ or $(\mathcal{U}_i := \mathcal{U}_{i-1} \cup \{q\} \wedge \mathcal{L}_i := \mathcal{L}_{i-1}))$. This time, we have $\varphi_i := \varphi_{i-1} - sB_q^m$ ($s = c_x$ or $s = C_x$) and, according to (5.19), we look for the optimal element $\psi_i^* \in \Pi_m^{\mathcal{F}_i}$. One can see clearly that

$$\left\{B_j^m \,:\, j \in \mathcal{F}_i\right\} = \{B_h^m \,:\, h \in \mathcal{F}_{i-1}\} \setminus \left\{B_q^m\right\}.$$

Now, since the dual basis for the basis $\{B_h^m : h \in \mathcal{F}_{i-1}\}$ of the space $\Pi_m^{\mathcal{F}_{i-1}}$ was computed in the previous iteration, we obtain the new dual basis using Algorithm 3.11 (cf. Case 1). Finally, we use Fact 3.2 to compute the optimal element.

Remark 5.9. Notice that φ_i $(i > 1)$ only *slightly* differs from φ_{i-1} (see Cases 1 and 2). Similarly as in Corollaries 3.5 and 3.10, one can obtain formulas connecting the coefficients of the new optimal element ψ_i^* and the previous one ψ_{i-1}^*.

5.6 Examples

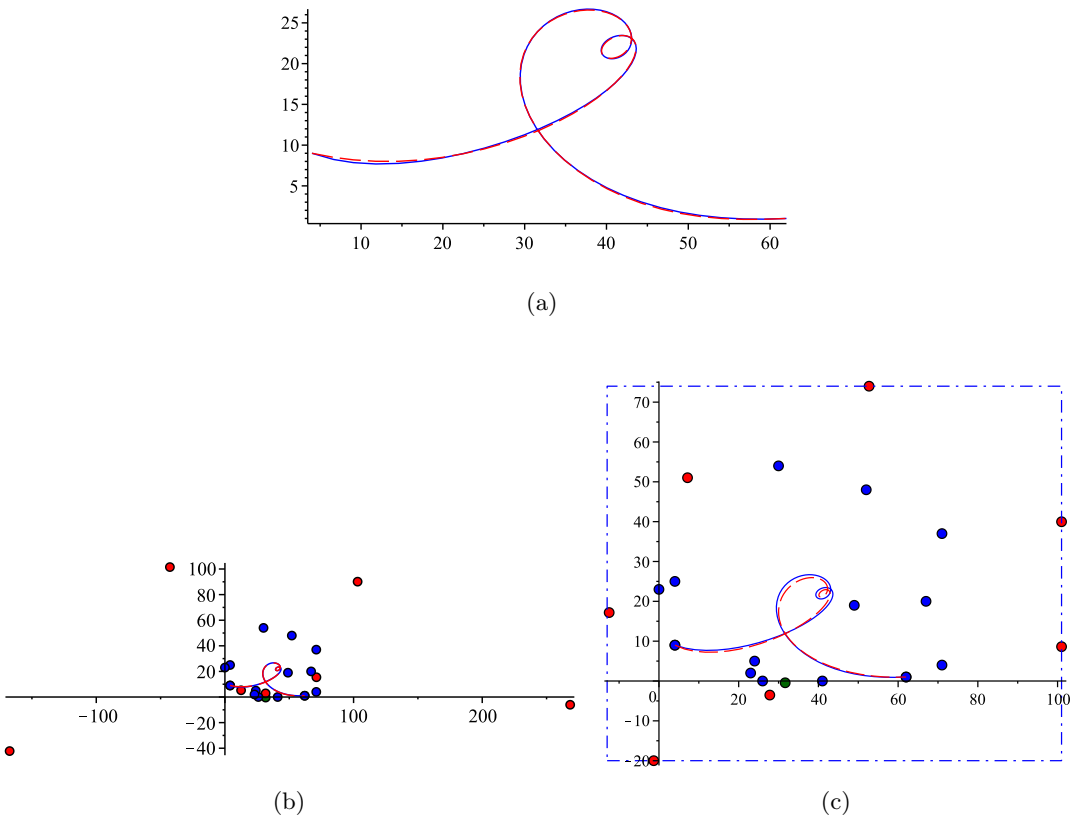
This subsection provides several examples of application of the discussed methods. The results have been obtained on a computer with `Intel Core i5-3337U 1.8GHz` processor and `8GB` of `RAM`, using Maple$^{\text{TM}}$13 with 16-digit arithmetic. An optimal solution of Problem 5.1 is computed by solving a system of normal equations, using Maple$^{\text{TM}}$ `fsolve` procedure. An optimal solution of Problem 5.5 is obtained using BVLS algorithm.

For each example, we give least squares error E (see (5.3)) and maximum error E_∞ (see (1.11)). In each case, we use the sequence $\{t_h\}_{h=0}^N$ of equally spaced points for the least squares distance (5.3), i.e., $t_h := h/N$ $(h = 0, 1, \ldots, N)$. According to the experiments, different choices of $\{t_k\}_{k=0}^N$ do not improve the results.

Example 5.10. First, let us apply the algorithms to the Bézier curve "Double loop" of degree 13. The control points are given in [92, Appendix B]. We set $k := 0$, $l := 1$, $m := 9$, $N := 18$ and

$$\begin{aligned} c_x &:= \min_{0\le i\le n} p_i^x - 13 = -13, \qquad & C_x &:= \max_{0\le i\le n} p_i^x + 30 = 101, \\ c_y &:= \min_{0\le i\le n} p_i^y - 20 = -20, \qquad & C_y &:= \max_{0\le i\le n} p_i^y + 20 = 74. \end{aligned} \tag{5.20}$$

Figures 14a–14c illustrate the results of degree reduction, i.e., the obtained Bézier curves of degree 9. Observe that the solution of Problem 5.1 is very precise (see Figure 14a; errors: $E = 4.19e{-}1$ and $E_\infty = 3.46e{-}1$). However, it has the drawback described in §5.1 (see Figure 14b), whereas the solution of Problem 5.5 is much more satisfying in this regard (see Figure 14c; errors: $E = 3.28e{+}0$ and $E_\infty = 1.43e{+}0$). Obviously, because of the additional restrictions, the errors for the box-constrained approach must be inevitably larger than in the case of the traditional degree reduction.



(a)

(b) (c)

Figure 14: The original Bézier curve of degree 13 (blue solid line with blue control points) reduced to the Bézier curve of degree 9 (red dashed line with red and green control points). Figure (a) shows the original curve and the curve being the optimal solution of Problem 5.1. Figure (b) illustrates the same curves as (a) but with the resulting control points. Figure (c) presents the curve being the optimal solution of Problem 5.5 with the resulting control points. The control points which are constrained by the continuity conditions are green, while the other ones are red and in the case of (c) bounded by the blue dotted-dashed frame (see (5.5) and (5.20)).

Example 5.11. Let there be given the composite Bézier curve "The Ugly Duckling sketch" (see Figures 15a and 15b) obtained by joining three Bézier curves of degrees 11, 5 and 11, respectively. For the control points, see [103, Example 6.2]. We look for Bézier curves of lower degrees 7, 4 and 7, respectively. Let us set $N := 14, 9, 14$, respectively; and $k = l := 0$ for each Bézier curve. The algorithms are applied independently to every segment of the composite curve.

Figure 15c shows the plots of the original curves and the corresponding results of the traditional degree reduction (see Problem 5.1). Once again, we obtain good approximations (errors: $E = 7.38e{+}0$, $3.01e{+}0$, $1.99e{+}0$, respectively; and $E_\infty = 5.28e{+}0$, $1.76e{+}0$, $2.27e{+}0$, respectively), however, the computed control points are located far away from the plots of the curves (see Figure 15d). To avoid this issue, we solve Problem 5.5 three times and obtain the curves shown in Figure 16a (errors: $E = 2.82e{+}1$, $2.20e{+}1$, $4.76e{+}1$, respectively; and $E_\infty = 2.46e{+}1$, $1.39e{+}1$, $2.83e{+}1$, respectively). The rectangular areas can be seen in Figures 16b–16d.

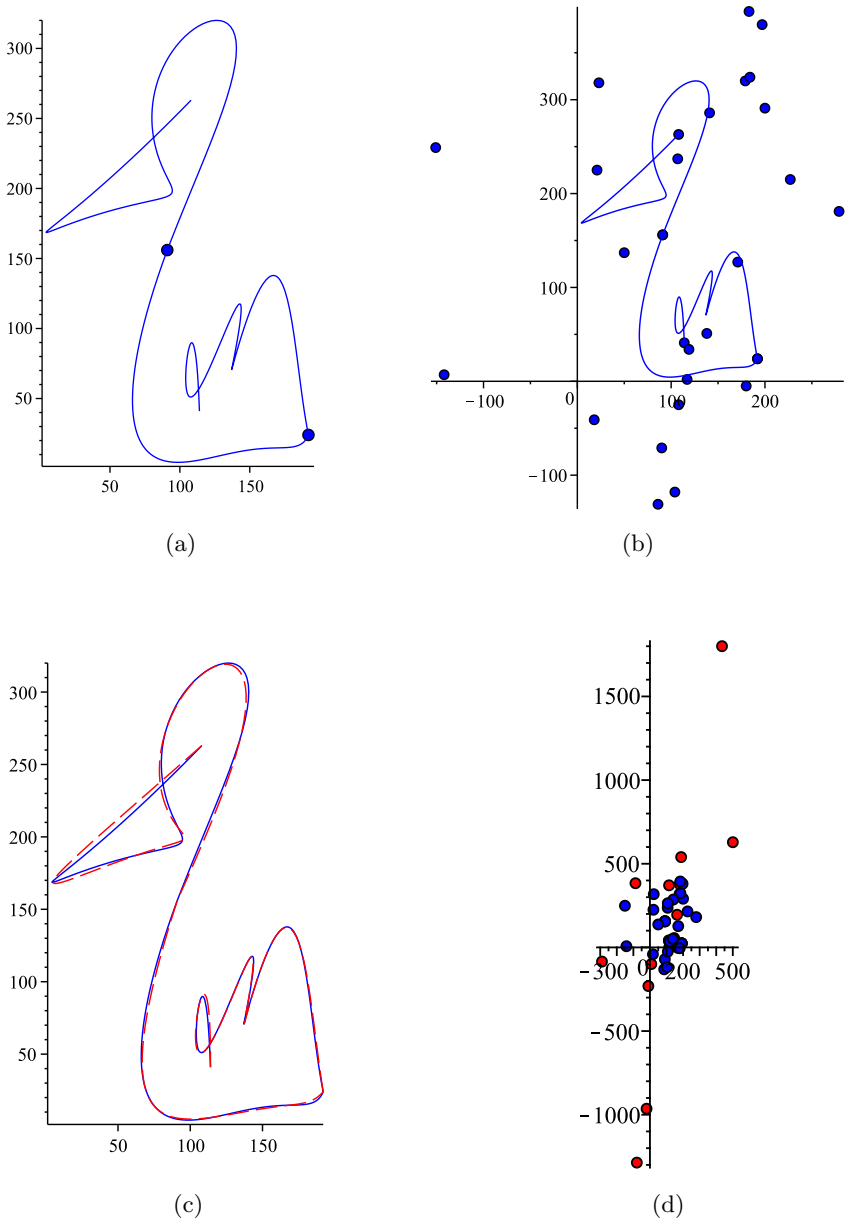


Figure 15: Figure (a) shows the original composite Bézier curve. Figure (b) presents the same composite Bézier curve as Figure (a) but with its control points. Figure (c) illustrates the original composite Bézier curve (blue solid line) and the degree reduced composite Bézier curve (red dashed line), where each segment is the optimal solution of Problem 5.1. Figure (d) shows the original control points which are blue, and the resulting ones which are red.

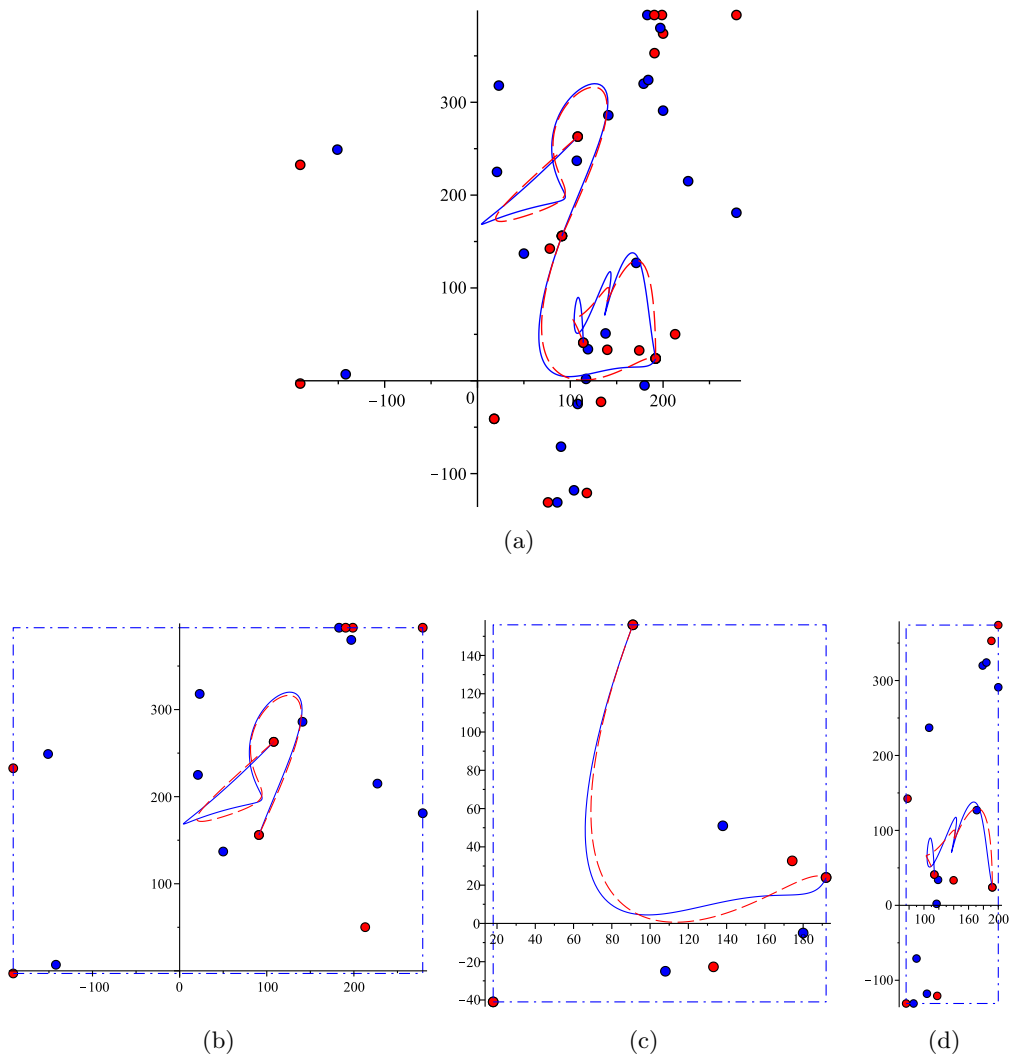


Figure 16: The original composite Bézier curve (blue solid line with blue control points) and the degree reduced composite Bézier curve (red dashed line with red control points), where each segment is the optimal solution of Problem 5.5. See the rectangular areas (blue dotted-dashed frames).

Example 5.12. Now, we consider the problem of degree reduction of sixteen-segment composite Bézier curve "Octopus" (see Figures 17a and 17b). The control points can be found at `http://www.ii.uni.wroc.pl/~pgo/octopus.txt`. We apply the algorithms independently to every segment of the composite curve. In Table 7, we give the parameters and errors.

As a result of the traditional degree reduction (see Problem 5.1), we obtain the composite curve with the control points shown in Figure 17c. Clearly, some of the control points are located far away from the plot of the curve (cf. Figure 17b). Next, let us focus on the box-constrained degree reduction (see Problem 5.5). For each resulting Bézier curve, the box constraints were chosen so that the searched control points are placed inside the rectangular area, bounded by the outermost control points of the original corresponding Bézier curve. More precisely, we set

$$c_z := \min_{0\le i\le n} p_i^z, \qquad C_z := \max_{0\le i\le n} p_i^z \qquad (z = x, y). \tag{5.21}$$

The resulting composite curve with its control points is illustrated in Figure 17d. This time, the location of the control points is much more satisfying (cf. Figure 17c). However, because of the additional restrictions (5.5), the larger errors are unavoidable (see Table 7).

In Table 8, we give the comparison of total running times between the traditional degree reduction and the box-constrained degree reduction using BVLS algorithm with and without the use of dual bases. The experiments show that BVLS algorithm combined with the method of solving the subproblem given in §5.5.2 is approximately two times faster than the one with the method from §5.5.1. However, the box-constrained degree reduction is still inevitably slower than the traditional degree reduction, which is much more simple.

Remark 5.13. Note that in [43], Wolfe's method [99] was also tested. However, according to [43, Table 1], it is less effective than BVLS algorithm combined with the straightforward method of solving the subproblem (see §5.5.1). Therefore, it is not worthy of consideration.

Remark 5.14. According to Remark 5.7, BVLS algorithm can be used to obtain an approximate solution of Problem 5.5 (see [43, Example 6.2]). However, thanks to the method given in §5.5.2, the cost of a single iteration of BVLS algorithm is now lower and such a strategy is no longer worthwhile.

	Input data						Problem 5.1		Problem 5.5	
	Curves	n	m	N	k	l	E	E_∞	E	E_∞
Octopus	Head: left side	9	7	20	2	1	$5.07e{-4}$	$2.30e{-4}$	$5.72e{-3}$	$2.32e{-3}$
	Head: right side	9	7	20	1	0	$9.05e{-5}$	$5.12e{-5}$	$2.29e{-3}$	$8.86e{-4}$
	1st arm: part 1	15	9	23	0	0	$2.65e{-4}$	$1.58e{-4}$	$3.53e{-3}$	$1.40e{-3}$
	1st arm: part 2	17	9	28	0	1	$1.59e{-3}$	$7.86e{-4}$	$1.87e{-3}$	$8.92e{-4}$
	2nd arm: part 1	14	10	26	1	0	$1.62e{-4}$	$9.51e{-5}$	$1.32e{-2}$	$4.41e{-3}$
	2nd arm: part 2	14	10	26	0	1	$7.35e{-5}$	$3.54e{-5}$	$4.44e{-3}$	$2.44e{-3}$
	3rd arm: part 1	13	7	25	1	1	$2.50e{-3}$	$9.56e{-4}$	$2.62e{-2}$	$9.03e{-3}$
	3rd arm: part 2	11	7	23	1	0	$8.73e{-4}$	$5.13e{-4}$	$4.05e{-3}$	$1.62e{-3}$
	4th arm	11	7	23	0	0	$2.78e{-3}$	$1.47e{-3}$	$2.61e{-2}$	$1.01e{-2}$
	5th arm	18	11	23	0	0	$1.51e{-4}$	$2.91e{-4}$	$6.14e{-3}$	$2.41e{-3}$
	6th arm: part 1	12	7	28	0	0	$1.44e{-3}$	$6.37e{-4}$	$6.97e{-3}$	$2.71e{-3}$
	6th arm: part 2	17	9	29	0	2	$8.48e{-4}$	$5.16e{-4}$	$1.39e{-2}$	$4.26e{-3}$
	7th arm: part 1	15	9	29	2	0	$9.86e{-4}$	$3.60e{-4}$	$4.39e{-3}$	$1.81e{-3}$
	7th arm: part 2	11	7	25	0	2	$8.95e{-4}$	$3.54e{-4}$	$8.95e{-4}$	$3.54e{-4}$
	8th arm: part 1	16	9	29	2	0	$1.21e{-3}$	$4.44e{-4}$	$1.38e{-2}$	$4.66e{-3}$
	8th arm: part 2	9	6	18	0	2	$2.29e{-3}$	$1.05e{-3}$	$2.29e{-3}$	$1.05e{-3}$

Table 7: The results of degree reduction of the composite Bézier curve "Octopus".

	Problem 5.1	Problem 5.5	
	System of normal equations	Without the use of dual bases	With the use of dual bases
Running times [s]	0.516	2.436	1.249

Table 8: Total running times of degree reduction of the composite Bézier curve "Octopus". For the parameters, see Table 7.

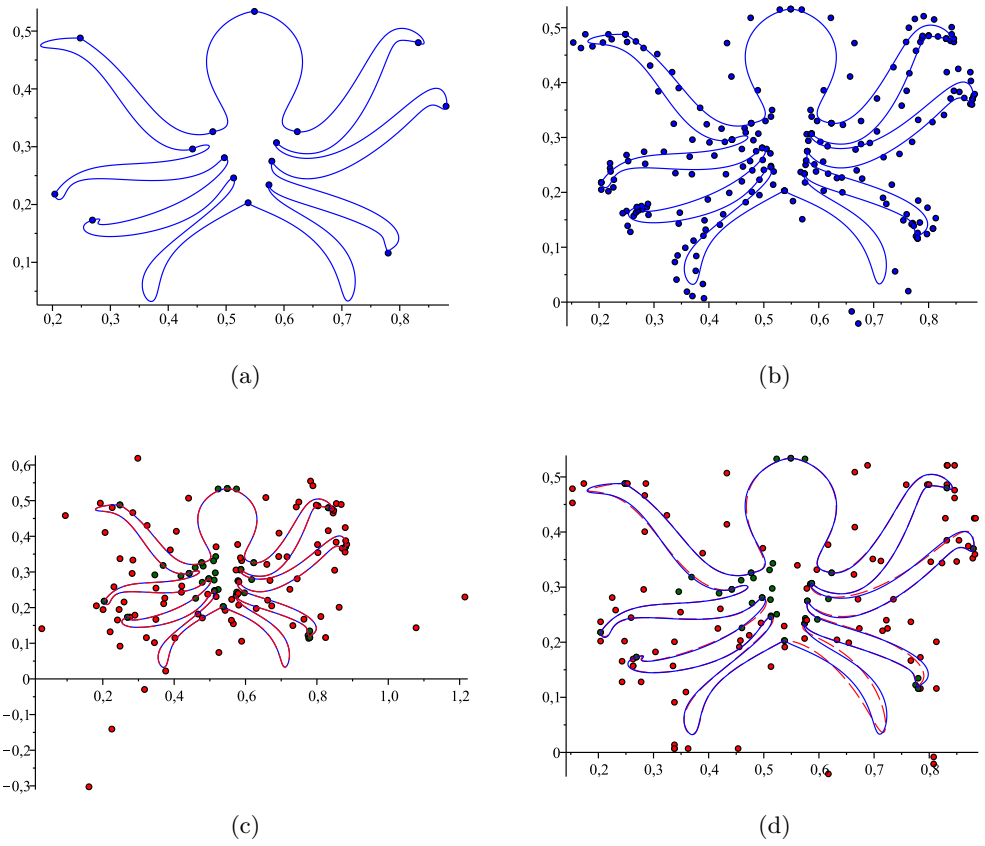


Figure 17: Figure (a) shows the original composite Bézier curve. Figure (b) presents the same composite Bézier curve as Figure (a) but with its control points. Figures (c) and (d) illustrate degree reduction of the original composite Bézier curve (blue solid line) to degree reduced composite Bézier curve (red dashed line with red and green control points), where each segment is (c) optimal solution of Problem 5.1, (d) optimal solution of Problem 5.5. The control points which are constrained by the continuity conditions (5.4) are green, while the other ones are red and restricted by (5.5). Parameters are specified in Table 7.

Remark 5.15. Now, let us discuss the issue of setting the rectangular area (5.5). A good way to start is to choose (5.21). Usually, this simple idea works quite well (see Example 5.12). In some cases we can even reduce the size of the rectangle, as in Example 5.4, where we set

$$\begin{aligned} c_x &:= \min_{0\le k\le n} p_k^x = -28, \qquad & C_x &:= \max_{0\le k\le n} p_k^x = 196, \\ c_y &:= \min_{0\le k\le n} p_k^y = -15, \qquad & C_y &:= \max_{0\le k\le n} p_k^y - 65 = 135, \end{aligned}$$

and the result is still good enough. However, Example 5.10 shows that sometimes we should expand the area to get satisfying results (see (5.20)). Clearly, the choice of the restrictions depends on the considered example. In general, this is a difficult problem to solve. See Remark 8.8, where one can find more detailed discussion on setting the rectangular area in the case of merging of planar Bézier curves with box constraints. Both issues are analogous.

Chapter 6

$C^{k,l}$-constrained merging of Bézier curves

In this chapter, we solve efficiently the problem of $C^{k,l}$-constrained merging of multiple segments of Bézier curves (see Problem 1.10). The novel method is based on the idea of using fast schemes of evaluation of certain connections involving Bernstein and dual Bernstein polynomials (see §3.3.1). The complexity of the algorithm is $O(sm^2)$, which is significantly less than complexity of other merging algorithms. For example, the one given in [72, §3.1] has the complexity $O(sm^3)$. In contrast to some other methods, we avoid matrix inversion. For a short description of available methods, see §1.6.2. This chapter is based on paper [102].

The outline of the chapter is as follows. In §6.1, which has preliminary character, we give efficient method of subdivision of Bézier curves. §6.2 brings a complete solution of Problem 1.10. In addition, it deals with algorithmic implementation of the proposed method. Some illustrative examples can be found in §6.3.

6.1 Efficient subdivision of Bézier curves

First of all, we have to establish the correspondence between P (1.21) and R (1.22). Taking into account that P is a composite Bézier curve, we have to subdivide the searched Bézier curve R as well. Thus, we will need the following restriction of the representation of B_j^m to a subinterval of the interval [0, 1] (cf. §1.4, pt. 9).

Lemma 6.1. *Let* $0 = t_0 < t_1 < \ldots < t_s = 1$ *be a partition of the interval* [0, 1]. *The following formula holds:*

$$B_j^m(t) = \sum_{h=0}^{m} d_{jh}^{(i)} B_h^m \left(\frac{t - t_{i-1}}{\Delta t_{i-1}} \right), \tag{6.1}$$

where

$$d_{jh}^{(i)} := \sum_{v=0}^{h} B_{j-v}^{m-h}(t_{i-1}) B_v^h(t_i).$$

Proof. The result is obtained in two steps. First, we subdivide the polynomial

$$B_j^m(t) = \sum_{h=0}^{m} \delta_{jh} B_h^m(t)$$

at the point t_i to get two forms for the subintervals $[0,\, t_i]$ and $[t_i,\, 1]$. Next, we subdivide the form corresponding to $[0,\, t_i]$ at t_{i-1}/t_i. We obtain the formula (6.1) with the coefficients $d_{jh}^{(i)}$ given by

$$d_{jh}^{(i)} := \sum_{w=0}^{m-h} B_w^{m-h}(t_{i-1}/t_i) B_j^{w+h}(t_i)$$

(we ignore the fact that the initial terms of the sum vanish as $B_j^{w+h}(t_i) = 0$ for $0 \le w < j-h$). Using the identity

$$B_j^{n+q}(x) = \sum_{w=0}^{q} B_w^q(x) B_{j-w}^n(x),$$

which can be easily proved using some basic properties of Bernstein polynomials (see, e.g., [33, §6.10]), and §1.3, pt. 10, it can be seen that

$$\begin{aligned} d_{jh}^{(i)} &= \sum_{w=0}^{m-h} B_w^{m-h}(t_{i-1}/t_i) \sum_{v=0}^{h} B_v^h(t_i) B_{j-v}^w(t_i) \\ &= \sum_{v=0}^{h} B_v^h(t_i) \sum_{w=0}^{m-h} B_w^{m-h}(t_{i-1}/t_i) B_{j-v}^w(t_i) \\ &= \sum_{v=0}^{h} B_v^h(t_i) B_{j-v}^{m-h}(t_{i-1}). \end{aligned}$$

□

Remark 6.2. Using Fact 3.1, we note that the formula (6.1) is equivalent to

$$B_j^m(u\Delta t_{i-1} + t_{i-1}) = \sum_{h=0}^{m} d_{jh}^{(i)} B_h^m(u) \qquad (u \in [0,\, 1]), \tag{6.2}$$

where

$$d_{jh}^{(i)} = \int_0^1 B_j^m(u\Delta t_{i-1} + t_{i-1}) D_h^m(u)\, \mathrm{d}u \tag{6.3}$$

with $D_h^m(u) \equiv D_h^m(u; 0, 0)$ being defined in §3.3

Lemma 6.3. *For $i = 1, 2, \ldots s$, the coefficients $d_{jh}^{(i)}$ satisfy the following recurrence equation:*

$$\begin{aligned} \Delta t_{i-1} \left[(m-j+1) d_{j-1,h}^{(i)} + (2j-m) d_{jh}^{(i)} - (j+1) d_{j+1,h}^{(i)} \right] \\ = (m-h) d_{j,h+1}^{(i)} + (2h-m) d_{jh}^{(i)} - h d_{j,h-1}^{(i)} \\ (1 \le j \le m-1;\ 0 \le h \le m). \end{aligned}$$

Proof. We differentiate both sides of the equation (6.2) with respect to u, and make use of the identity

$$\frac{\mathrm{d}}{\mathrm{d}u} B_j^m(u) = (m-j+1) B_{j-1}^m(u) + (2j-m) B_j^m(u) - (j+1) B_{j+1}^m(u).$$

Equating the Bézier coefficients gives the result. □

Lemma 6.3 yields the following algorithm of computing the coefficients $d_{jh}^{(i)}$ $(i = 1, 2, \ldots, s;$ $j, h = 0, 1, \ldots, m)$.

Algorithm 6.4. *[Computing the coefficients $d_{jh}^{(i)}$]*
Input: *m, s, $0 = t_0 < t_1 < \ldots < t_s = 1$*
Output: *table of the coefficients $d_{jh}^{(i)}$ $(i = 1, 2, \ldots, s;\ j, h = 0, 1, \ldots, m)$*

`Step 1.` *For $i = 1, 2, \ldots, s$, compute*

$$d_{-1,0}^{(i)} := 0, \quad d_{00}^{(i)} := (1 - t_{i-1})^m,$$
$$d_{-1,h}^{(i)} := 0, \quad d_{0h}^{(i)} := \frac{1 - t_i}{1 - t_{i-1}} d_{0,h-1}^{(i)} \qquad (h = 1, 2, \ldots, m).$$

`Step 2.` *For $i = 1, 2, \ldots, s$; $j = 0, 1, \ldots, m-1$; $h = 0, 1, \ldots, m$, compute*

$$\begin{aligned} d_{j+1,h}^{(i)} :=& (j+1)^{-1} \Big\{ (\Delta t_{i-1})^{-1} \left[h d_{j,h-1}^{(i)} - (2h - m) d_{jh}^{(i)} - (m - h) d_{j,h+1}^{(i)} \right] \\ & + (m - j + 1) d_{j-1,h}^{(i)} + (2j - m) d_{jh}^{(i)} \Big\}. \end{aligned}$$

Observe that the complexity of Algorithm 6.4 is $O(sm^2)$.

6.2 Solution and algorithm

Recall that, in §2.1, we have related the $C^{k,l}$ continuity conditions (1.24) with the control points of the curves P^1, P^s and R. Consequently, $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$ can be computed using the formulas (2.5) and (2.6), respectively. The remaining control points $r_{k+1}, r_{k+2}, \ldots, r_{m-l-1}$ are to be determined so that the L_2-error (1.23) is minimized. Taking into account that Problem 1.10 can be solved in a componentwise way, we can limit ourselves to the case of $d = 1$. Hence, it is sufficient to deal with the following problem (cf. Problem 1.10).

Problem 6.5. *[Merging of Bézier curves with prescribed boundary control points]*
Let $0 = t_0 < t_1 < \ldots < t_s = 1$ be a partition of the interval $[0, 1]$. Given a piecewise polynomial function P with segments $P^1, P^2, \ldots, P^s$ such that

$$P(t) = P^i(t) := \sum_{j=0}^{n_i} p_j^i \, B_j^{n_i} \left(\frac{t - t_{i-1}}{\Delta t_{i-1}} \right) \qquad (t \in [t_{i-1},\, t_i];\ i = 1, 2, \ldots, s), \tag{6.4}$$

we look for a polynomial R of degree m,

$$R(t) = \sum_{j=0}^{m} r_j \, B_j^m(t) \qquad (t \in [0,\, 1]), \tag{6.5}$$

having the prescribed coefficients $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$, which gives the minimum value of the L_2-error

$$\|P - R\|_{L_2} := \sqrt{\langle P - R, P - R \rangle_L} \tag{6.6}$$

(cf. (3.16)*).*

The following theorem deals with Problem 6.5.

Theorem 6.6. *Let there be given the coefficients p_j^i ($i = 1, 2, \ldots, s$; $j = 0, 1, \ldots, n_i$) of the polynomials* (6.4)*, and the coefficients r_j ($j = 0, 1, \ldots, k, m-l, m-l+1, \ldots, m$) of the polynomial* (6.5)*. The* inner *coefficients r_j ($j = k+1, k+2, \ldots, m-l-1$) of the polynomial* (6.5) *minimizing the L_2-error* (6.6) *are given by*

$$r_j = \sum_{h=k+1}^{m-l-1} \hat{r}_h c_{hj}(m,k,l) \qquad (j = k+1, k+2, \ldots, m-l-1), \tag{6.7}$$

where

$$\hat{r}_h := \sum_{i=1}^{s} \Delta t_{i-1} \sum_{v=0}^{m} \hat{p}_v^i d_{hv}^{(i)} - \frac{1}{2m+1}\binom{m}{h}\left(\sum_{v=0}^{k} + \sum_{v=m-l}^{m}\right)\binom{2m}{h+v}^{-1}\binom{m}{v} r_v, \tag{6.8}$$

$$\hat{p}_v^i := \frac{1}{m+n_i+1}\binom{m}{v}\sum_{q=0}^{n_i}\binom{m+n_i}{q+v}^{-1}\binom{n_i}{q} p_q^i \tag{6.9}$$

with $c_{hj}(m,k,l)$ and $d_{jh}^{(i)}$ being introduced in §3.3.1 and §6.1, respectively.

Proof. First, using Lemma 3.13, we represent each segment P^i of the original piecewise polynomial P in the dual Bernstein basis of degree m,

$$P^i(t) = \sum_{v=0}^{m} \hat{p}_v^i D_v^m\left(\frac{t - t_{i-1}}{\Delta t_{i-1}}\right)$$

with $\hat{p}_v^i$ being defined in (6.9).

Next, we note that

$$\|P - R\|_{L_2}^2 = \|W - S\|_{L_2}^2,$$

where

$$W := P - \left(\sum_{h=0}^{k} + \sum_{h=m-l}^{m}\right) r_h B_h^m, \qquad S := \sum_{j=k+1}^{m-l-1} r_j B_j^m.$$

Now, the goal is to obtain the coefficients $\hat{r}_j$ of the searched polynomial in the constrained dual Bernstein form,

$$S = \sum_{j=k+1}^{m-l-1} \hat{r}_j D_j^{(m,k,l)}$$

(cf. §3.3). Then, the Bézier coefficients r_j of S can be easily computed using (6.7) (cf. Lemma 3.14). Using Fact 3.2 and (6.3), we obtain

$$\begin{aligned}
\hat{r}_j &= \langle W, B_j^m \rangle_L = \int_0^1 W(t) B_j^m(t)\,\mathrm{d}t \\
&= \sum_{i=1}^{s}\sum_{h=0}^{m} \hat{p}_h^i \int_{t_{i-1}}^{t_i} D_h^m\left(\frac{t - t_{i-1}}{\Delta t_{i-1}}\right) B_j^m(t)\,\mathrm{d}t \\
&\qquad - \left(\sum_{h=0}^{k} + \sum_{h=m-l}^{m}\right) r_h \int_0^1 B_h^m(t) B_j^m(t)\,\mathrm{d}t
\end{aligned}$$

$$
\begin{aligned}
=&\sum_{i=1}^{s}\sum_{h=0}^{m}\hat{p}_h^i\Delta t_{i-1}\int_0^1 D_h^m(u)B_j^m(u\Delta t_{i-1}+t_{i-1})\,\mathrm{d}u\\
&\qquad-\left(\sum_{h=0}^{k}+\sum_{h=m-l}^{m}\right)r_h\frac{1}{2m+1}\binom{m}{h}\binom{m}{j}\binom{2m}{h+j}^{-1}\\
=&\sum_{i=1}^{s}\Delta t_{i-1}\sum_{h=0}^{m}\hat{p}_h^i d_{jh}^{(i)}-\frac{1}{2m+1}\binom{m}{j}\left(\sum_{h=0}^{k}+\sum_{h=m-l}^{m}\right)r_h\binom{m}{h}\binom{2m}{h+j}^{-1}\\
&\qquad\qquad(j=k+1,k+2,\ldots,m-l-1).
\end{aligned}
$$

This completes the proof. □

Now, let the composite curve P and the merged curve R be the curves in $\mathbb{R}^d$ $(d\geq 1)$. Let $p_j^i=(p_{j1}^i,p_{j2}^i,\ldots,p_{jd}^i)$ $(i=1,2,\ldots,s;\ j=0,1,\ldots,n_i)$, and $r_j=(r_{j1},r_{j2},\ldots,r_{jd})$ $(j=0,1,\ldots,m)$ be the control points of P and R, respectively. For $i=1,2,\ldots,s$; $h=1,2,\ldots,d$, we define vectors

$$\pi_h^i:=\left[p_{0h}^i,p_{1h}^i,\ldots,p_{n_i,h}^i\right]\in\mathbb{R}^{n_i+1},\tag{6.10}$$

$$\varrho_h^i:=\left[\varrho_{0h}^i,\varrho_{1h}^i,\ldots,\varrho_{mh}^i\right]\in\mathbb{R}^{m+1},\tag{6.11}$$

where

$$\varrho_{zh}^i:=\sum_{j=0}^{m}r_{jh}d_{jz}^{(i)}\qquad(z=0,1,\ldots,m).\tag{6.12}$$

It is easy to see that the L_2-distance between P and R can be written in the following way:

$$
\begin{aligned}
E_2:=&\|P-R\|_{L_2}\\
=&\sqrt{\sum_{i=1}^{s}\Delta t_{i-1}\sum_{h=1}^{d}\left[J_{n_i,n_i}(\pi_h^i,\pi_h^i)-2J_{n_i,m}(\pi_h^i,\varrho_h^i)+J_{mm}(\varrho_h^i,\varrho_h^i)\right]},
\end{aligned}\tag{6.13}
$$

where

$$J_{NM}(u,v):=\sum_{j=0}^{N}u_j\sum_{z=0}^{M}a_{jz}^{(N,M)}v_z\tag{6.14}$$

with $u=[u_0,u_1,\ldots,u_N]$ and $v=[v_0,v_1,\ldots,v_M]$. Here we use the notation of (3.18).

Remark 6.7. Note that in contrast to Problem 1.9, we are dealing with the L_2-error (6.13) instead of the weighted L_2-error (1.13). The weight function would significantly complicate the problem of merging and the possible improvement of results would be negligible.

Now, we present the algorithm of solving Problem 1.10.

Algorithm 6.8. *[$C^{k,l}$-constrained merging of Bézier curves]*

Input: *n_i $(i=1,2,\ldots,s)$ – degrees of the curves (1.21);*

p_j^i $(j=0,1,\ldots,n_i;\ i=1,2,\ldots,s)$ – control points of the curves (1.21);

m – degree of the curve (1.22);

k, l – orders of the parametric continuity (see (1.24));

$0 = t_0 < t_1 < \ldots < t_s = 1$ – *partition of the interval* [0, 1]
Assumptions: $m \geq \max_i n_i$; $k \leq n_1$; $l \leq n_s$; $k, l \geq -1$; $k + l < m - 1$
Output: *control points of the* $C^{k,l}$*-constrained merged Bézier curve, and error* E_2

Step 1. *Compute* $r_0, r_1, \ldots, r_k$ *by* (2.5).

Step 2. *Compute* $r_m, r_{m-1}, \ldots, r_{m-l}$ *by* (2.6).

Step 3. *Compute* $\hat{p}_j^i$ $(i = 1, 2, \ldots, s;\ j = 0, 1, \ldots, m)$ *by* (6.9).

Step 4. *Compute* $d_{jh}^{(i)}$ $(i = 1, 2, \ldots, s;\ j, h = 0, 1, \ldots, m)$ *using Algorithm 6.4.*

Step 5. *Compute* $\hat{r}_j$ $(j = k + 1, k + 2, \ldots, m - l - 1)$ *by* (6.8).

Step 6. *Compute* $c_{ij}(m, k, l)$ $(i, j = k + 1, k + 2, \ldots, m - l - 1)$ *using Algorithm 3.15.*

Step 7. *Compute* r_j $(j = k + 1, k + 2, \ldots, m - l - 1)$ *by* (6.7).

Step 8. *Compute* ϱ_{zh}^i $(i = 1, 2, \ldots, s;\ z = 0, 1, \ldots, m;\ h = 1, 2, \ldots, d)$ *by* (6.12).

Step 9. *Compute* E_2 *by* (6.13).

Step 10. *Return* $r_0, r_1, \ldots, r_m$, *and* E_2.

Notice that the complexity of Algorithm 6.8 is $O(sm^2)$.

6.3 Examples

In this subsection, we give several examples of application of Algorithm 6.8. For each example, we give L_2-error E_2 (6.13) as well as maximum error E_∞ (1.11). The results have been obtained on a computer with `Intel Core i5-3337U 1.8GHz` processor and `8GB` of `RAM`, using Maple$^{\text{TM}}$13 with 32-digit arithmetic.

Generalizing the approach of [71, (6.1)], partition of the interval $[t_0, t_s] = [0, 1]$ is determined according to the lengths of segments P^i,

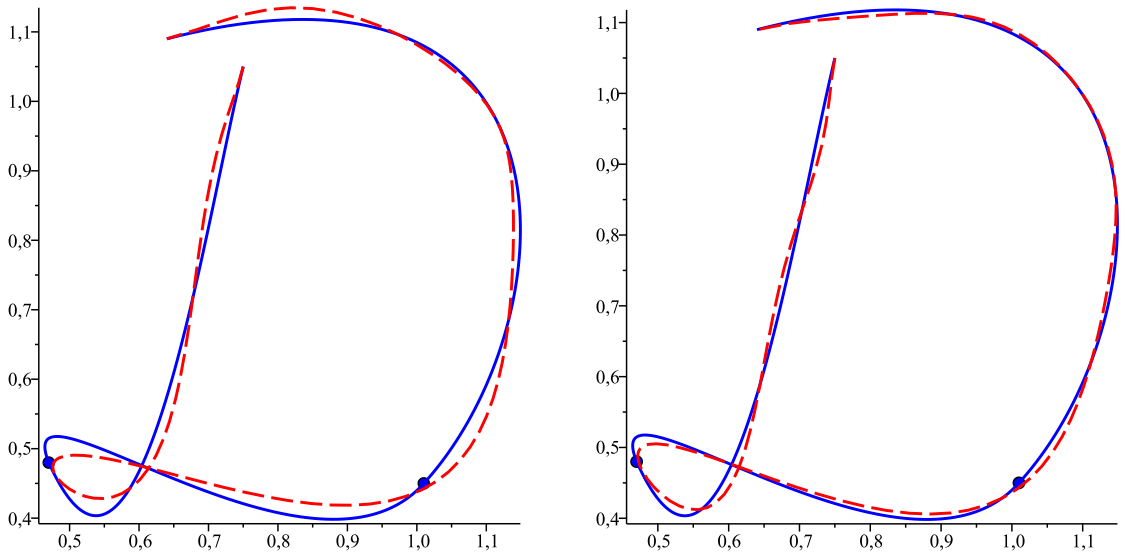
$$t_j := L_j / L_s \qquad (j = 1, 2, \ldots, s - 1), \tag{6.15}$$

where

$$L_q := \sum_{i=1}^{q} \int_0^1 \left\| \frac{\mathrm{d}}{\mathrm{d}t} \sum_{h=0}^{n_i} p_h^i B_h^{n_i}(t) \right\| \mathrm{d}t.$$

Integrals are evaluated using the Maple$^{\text{TM}}$13 function `int` with the option `numeric`.

Example 6.9. First, we recall the composite Bézier curve "D". For the control points, see Example 1.3. The formula (6.15) implies the following partition: $t_0 = 0$, $t_1 \doteq 0.32$, $t_2 \doteq 0.56$, $t_3 = 1$. The results of $C^{k,l}$-constrained merging are presented in Table 9. The original composite Bézier curve and two of the merged curves are plotted in Figures 18a and 18b.



(a) (b)

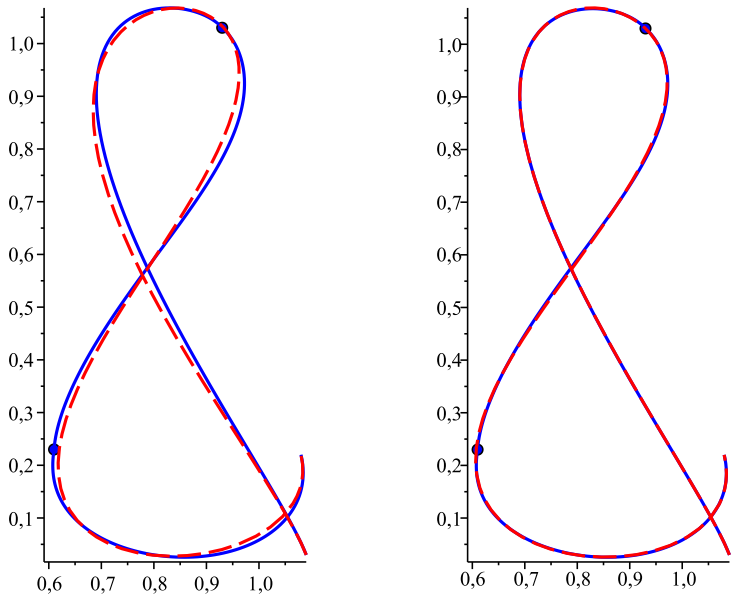
Figure 18: $C^{k,l}$-constrained merging of three segments of the composite Bézier curve. The original composite curve (blue solid line) and the merged curve (red dashed line) with parameters: (a) $m = 11$, $k = l = 1$; (b) $m = 13$, $k = l = 1$.

Parameters			Errors	
m	k	l	E_2	E_∞
11	0	0	$1.45e{-2}$	$3.09e{-2}$
	1	1	$1.67e{-2}$	$3.35e{-2}$
	2	2	$2.12e{-2}$	$4.22e{-2}$
12	0	0	$7.93e{-3}$	$2.00e{-2}$
	1	1	$9.10e{-3}$	$2.27e{-2}$
	2	2	$1.18e{-2}$	$2.92e{-2}$
13	0	0	$7.78e{-3}$	$2.06e{-2}$
	1	1	$9.05e{-3}$	$2.30e{-2}$
	2	2	$1.17e{-2}$	$2.75e{-2}$

Table 9: L_2-errors and maximum errors of the $C^{k,l}$-constrained merging of three segments of the composite Bézier curve "D".

Example 6.10. Now, we use Algorithm 6.8 to merge the composite Bézier curve "Ampersand", with three 5th degree Bézier segments, defined by the control points $\{(1.09, 0.03)$, $(1.02, 0.21)$, $(0.6, 0.75)$, $(0.5, 1.11)$, $(0.85, 1.12)$, $(0.93, 1.03)\}$, $\{(0.93, 1.03)$, $(1.01, 0.96)$, $(1.02,$ $0.76)$, $(0.8,\ 0.65)$, $(0.62,\ 0.38)$, $(0.61, 0.23)\}$, and $\{(0.61, 0.23)$, $(0.59, 0.1)$, $(0.67, 0.02)$, $(0.91,$ $-0.05)$, $(1.12,\ 0.05)$, $(1.08,\ 0.22)\}$, respectively (cf. Example 5.4). According to (6.15), we have $t_0 = 0$, $t_1 \doteq 0.45$, $t_2 \doteq 0.76$, $t_3 = 1$. The results are given in Table 10. Moreover, we give the running times required to compute the resulting control points in the case of Algorithm 6.8 and the one from [72, §3.1]. Clearly, Algorithm 6.8 is faster. Figures 19a and 19b

illustrate the results for two representative cases. This example shows that merging may result in data compression.



(a) (b)

Figure 19: $C^{k,l}$-constrained merging of three segments of the composite Bézier curve. The original composite curve (blue solid line) and the merged curve (red dashed line) with parameters: (a) $m = 8$, $k = 2$, $l = 1$; (b) $m = 12$, $k = 2$, $l = 1$.

Parameters			Errors		Running times [ms]	
m	k	l	E_2	E_∞	Algorithm 6.8	Lu [72, §3.1]
8	1	0	$4.82e{-3}$	$8.81e{-3}$	10	59
	1	1	$5.91e{-3}$	$1.13e{-2}$	11	60
	2	1	$1.06e{-2}$	$1.81e{-2}$	10	55
10	1	0	$1.71e{-3}$	$5.47e{-3}$	16	91
	1	1	$1.74e{-3}$	$5.35e{-3}$	15	84
	2	1	$1.83e{-3}$	$5.35e{-3}$	15	76
12	1	0	$1.66e{-3}$	$5.55e{-3}$	22	127
	1	1	$1.66e{-3}$	$5.55e{-3}$	22	125
	2	1	$1.69e{-3}$	$5.59e{-3}$	19	111

Table 10: L_2-errors, maximum errors and running times of the $C^{k,l}$-constrained merging of three segments of the composite Bézier curve "Ampersand".

Example 6.11. The curve "Penguin" is formed by two composite Bézier curves. The left curve has four cubic segments with the control points $\{(0.31, 0.23), (0.35, 0.19), (0.39, 0.23), (0.37, 0.26)\}$, $\{(0.37, 0.26), (0.21, 0.54), (0.53, 0.77), (0.21, 0.76)\}$, $\{(0.21, 0.76), (0.1, 0.76), (0.5, 0.88), (0.42, 0.79)\}$, and $\{(0.42, 0.79), (0.26, 0.76), (0.23, 0.92), (0.34, 0.94)\}$, respectively. The right one is composed of three cubic segments having the control points $\{(0.34, 0.94), (0.74, 0.99), (0.67, 0.19), (0.56, 0.21)\}$, $\{(0.56, 0.21), (0.19, 0.32), (0.62, 1.05), (0.56, 0.61)\}$, and $\{(0.56, 0.61), (0.5, 0.24), (0.41, 0.41), (0.5, 0.64)\}$, respectively. The formula (6.15) gives $t_0 = 0,\ t_1 \doteq 0.08,\ t_2 \doteq 0.55,\ t_3 \doteq 0.78,\ t_4 = 1$ for the left curve, and $t_0 = 0,\ t_1 \doteq 0.42,\ t_2 \doteq 0.78,\ t_3 = 1$ for the right one. The results of separate merging of segments of both curves can be seen in Table 11. Two selected cases are shown in Figures 20a and 20b.

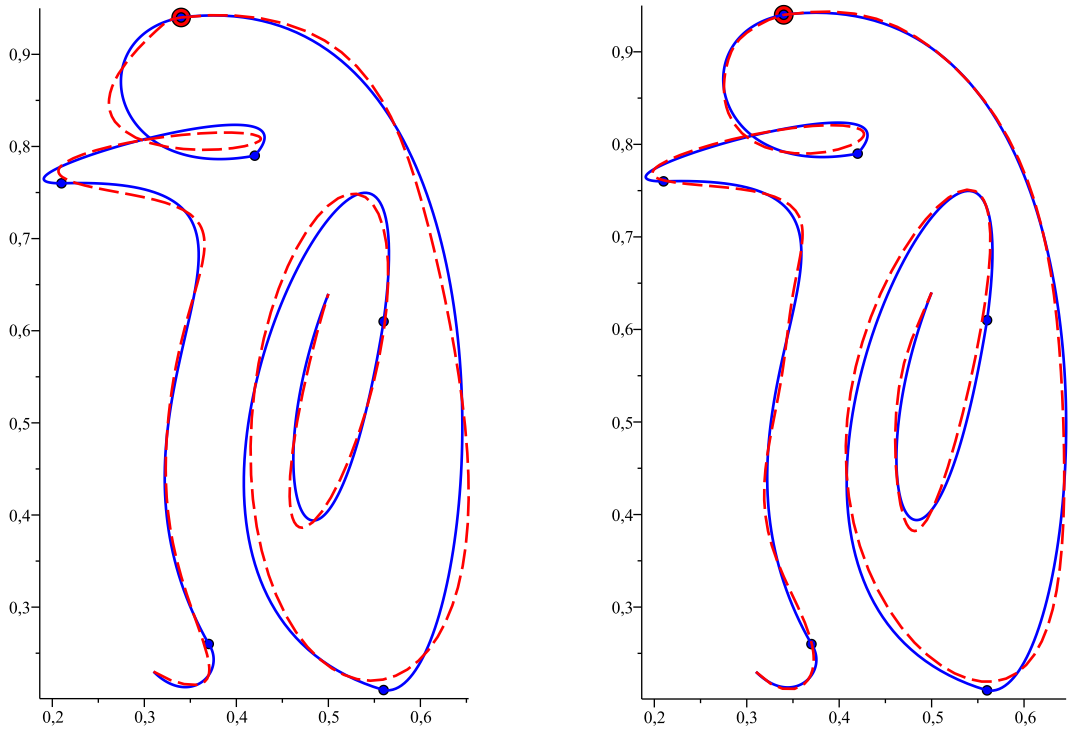


Figure 20: Separate $C^{k,l}$-constrained merging of segments of two composite Bézier curves. The original composite curves (blue solid line) and the merged curves (red dashed line). (a) Left curve: $m = 12$, $k = 0$, $l = 1$; right curve: $m = 10$, $k = 1$, $l = 0$. (b) Left curve: $m = 14$, $k = l = 1$; right curve: $m = 13$, $k = l = 1$.

Left curve					Right curve				
m	k	l	E_2	E_∞	m	k	l	E_2	E_∞
12	0	0	$7.45e{-3}$	$1.90e{-2}$	10	0	0	$1.28e{-2}$	$3.51e{-2}$
	0	1	$9.33e{-3}$	$2.08e{-2}$		1	0	$1.33e{-2}$	$3.67e{-2}$
	1	0	$7.51e{-3}$	$1.84e{-2}$		0	1	$1.37e{-2}$	$3.49e{-2}$
	1	1	$9.36e{-3}$	$2.12e{-2}$		1	1	$1.45e{-2}$	$3.69e{-2}$
13	0	0	$6.68e{-3}$	$1.45e{-2}$	12	0	0	$9.01e{-3}$	$3.00e{-2}$
	0	1	$7.15e{-3}$	$1.57e{-2}$		1	0	$9.46e{-3}$	$3.15e{-2}$
	1	0	$7.16e{-3}$	$1.50e{-2}$		0	1	$9.49e{-3}$	$2.99e{-2}$
	1	1	$7.83e{-3}$	$1.66e{-2}$		1	1	$9.78e{-3}$	$3.13e{-2}$
14	0	0	$4.39e{-3}$	$1.19e{-2}$	13	0	0	$8.65e{-3}$	$2.83e{-2}$
	0	1	$4.52e{-3}$	$1.21e{-2}$		1	0	$8.71e{-3}$	$2.82e{-2}$
	1	0	$4.49e{-3}$	$1.19e{-2}$		0	1	$9.47e{-3}$	$2.95e{-2}$
	1	1	$4.58e{-3}$	$1.21e{-2}$		1	1	$9.62e{-3}$	$2.95e{-2}$

Table 11: L_2-errors and maximum errors of separate $C^{k,l}$-constrained merging of segments of two composite Bézier curves “Penguin”.

Chapter 7

$G^{k,l}$-constrained merging of Bézier curves

In this chapter, we generalize the result from Chapter 6 to solve efficiently the problem of $G^{k,l}$-constrained merging of multiple segments of Bézier curves (see Problem 1.11). As in the case of the $G^{k,l}$-constrained degree reduction (see Problem 1.9), we are dealing with the problem in two-phases. The first phase is to compute optimal values of the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$ (see §2.2). Two different approaches, with and without the simplifying assumptions (see §2.3), are presented in §7.1. The second phase, where we compute the searched control points, is based on Theorem 6.6. The algorithms are given in §7.2. According to some numerical experiments, the approximation is much more accurate than that of Algorithm 6.8 (see §7.3). The material presented in this chapter is the author's independent work and it has not been published before.

First of all, notice that the formulas (2.17)–(2.22) with fixed parameters $\{\lambda_i\}$ and $\{\mu_j\}$ constitute constraints of the form demanded in Theorem 6.6. Consequently, the *inner* control points $r_{k+1}, r_{k+2}, \ldots, r_{m-l-1}$ of the curve (1.26) depend on these parameters. Because of this dependency, the result cannot be obtained in a componentwise way. Therefore, a generalization of Theorem 6.6 for any natural d should be used. This modification is very straightforward, thus it is omitted in this thesis. In the next subsection, we focus on computing the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$.

7.1 Computing the continuity parameters

Analogously as in §4.2, optimum values of the parameters $\{\lambda_i\}$ and $\{\mu_j\}$ can be obtained by minimizing the error function

$$\begin{aligned} E \equiv E(\lambda_1, \lambda_2, \ldots, \lambda_k, \mu_1, \mu_2, \ldots, \mu_l) &:= \int_0^1 \|P(t) - R(t)\|^2 \,\mathrm{d}t \\ &= \sum_{i=1}^{s} \Delta t_{i-1} \sum_{h=1}^{d} \left[J_{n_i,n_i}(\pi_h^i, \pi_h^i) - 2J_{n_i,m}(\pi_h^i, \varrho_h^i) + J_{mm}(\varrho_h^i, \varrho_h^i) \right], \end{aligned} \tag{7.1}$$

where J_{NM}, π_h^i, ϱ_h^i are given by (6.14), (6.10), (6.11), respectively. The function depends on these parameters *via* formulas (2.17)–(2.22) and (6.7). For a minimum of the function, it is

necessary that its derivatives with respect to the parameters are zero. As a result, we obtain the system

$$\left.\begin{aligned}\frac{\partial E}{\partial \lambda_u} &= \sum_{i=1}^{s} \Delta t_{i-1} \sum_{h=1}^{d} \left[J_{mm}\left(\varrho_h^i, \sigma_h^{(i,u)}\right) - J_{n_i,m}\left(\pi_h^i, \sigma_h^{(i,u)}\right)\right] = 0 \qquad (u = 1, 2, \ldots, k),\\ \frac{\partial E}{\partial \mu_v} &= \sum_{i=1}^{s} \Delta t_{i-1} \sum_{h=1}^{d} \left[J_{mm}\left(\varrho_h^i, \tau_h^{(i,v)}\right) - J_{n_i,m}\left(\pi_h^i, \tau_h^{(i,v)}\right)\right] = 0 \qquad (v = 1, 2, \ldots, l),\end{aligned}\right\} \tag{7.2}$$

where

$$\begin{aligned}\sigma_h^{(i,u)} &:= \left[\sigma_{0h}^{(i,u)}, \sigma_{1h}^{(i,u)}, \ldots, \sigma_{mh}^{(i,u)}\right] \in \mathbb{R}^{m+1},\\ \tau_h^{(i,u)} &:= \left[\tau_{0h}^{(i,v)}, \tau_{1h}^{(i,v)}, \ldots, \tau_{mh}^{(i,v)}\right] \in \mathbb{R}^{m+1}\end{aligned}$$

for

$$\sigma_{zh}^{(i,u)} := \sum_{j=u}^{m-l-1} d_{jz}^{(i)} \frac{\partial r_{jh}}{\partial \lambda_u}, \qquad \tau_{zh}^{(i,v)} := \sum_{j=k+1}^{m-v} d_{jz}^{(i)} \frac{\partial r_{jh}}{\partial \mu_v} \qquad (z = 0, 1, \ldots, m)$$

with $d_{jz}^{(i)}$ being introduced in §6.1.

Proceeding as in §4.2, we compute the partial derivatives of hth coordinates of the control points (2.17)–(2.22). In the case of $k = l = 3$, we obtain

$$\frac{\partial r_{ih}}{\partial \lambda_1} = \begin{cases} \frac{n_1}{m} t_1^{-1} \Delta p_{0h}^1 & (i = 1),\\ 2\frac{n_1}{m} t_1^{-1} \Delta p_{0h}^1 + 2\lambda_1 \frac{(n_1-1)_2}{(m-1)_2} t_1^{-2} \Delta^2 p_{0h}^1 & (i = 2),\\ 3\frac{n_1}{m} t_1^{-1} \Delta p_{0h}^1 + \left[2\lambda_1 + \frac{1}{m-2}\lambda_2\right] 3\frac{(n_1-1)_2}{(m-1)_2} t_1^{-2} \Delta^2 p_{0h}^1 + 3\lambda_1^2 \frac{(n_1-2)_3}{(m-2)_3} t_1^{-3} \Delta^3 p_{0h}^1 & (i = 3),\\ 0 & (i = 0;\ m-3 \le i \le m),\end{cases} \tag{7.3}$$

$$\frac{\partial r_{ih}}{\partial \lambda_2} = \begin{cases} \frac{n_1}{(m-1)_2} t_1^{-1} \Delta p_{0h}^1 & (i = 2),\\ 3\frac{n_1}{(m-1)_2} t_1^{-1} \Delta p_{0h}^1 + 3\lambda_1 \frac{(n_1-1)_2}{(m-2)_3} t_1^{-2} \Delta^2 p_{0h}^1 & (i = 3),\\ 0 & (i = 0, 1;\ m-3 \le i \le m),\end{cases} \tag{7.4}$$

$$\frac{\partial r_{ih}}{\partial \lambda_3} = \begin{cases} \frac{n_1}{(m-2)_3} t_1^{-1} \Delta p_{0h}^1 & (i = 3),\\ 0 & (i = 0, 1, 2;\ m-3 \le i \le m),\end{cases} \tag{7.5}$$

$$\frac{\partial r_{ih}}{\partial \mu_1} = \begin{cases} -\frac{n_s}{m}(1 - t_{s-1})^{-1} \Delta p_{n_s-1,h}^s & (i = m-1),\\ -2\frac{n_s}{m}(1 - t_{s-1})^{-1} \Delta p_{n_s-1,h}^s + 2\mu_1 \frac{(n_s-1)_2}{(m-1)_2}(1 - t_{s-1})^{-2} \Delta^2 p_{n_s-2,h}^s & (i = m-2),\\ -3\frac{n_s}{m}(1 - t_{s-1})^{-1} \Delta p_{n_s-1,h}^s + \left[2\mu_1 - \frac{1}{m-2}\mu_2\right] 3\frac{(n_s-1)_2}{(m-1)_2}(1 - t_{s-1})^{-2} \Delta^2 p_{n_s-2,h}^s & \\ -3\mu_1^2 \frac{(n_s-2)_3}{(m-2)_3}(1 - t_{s-1})^{-3} \Delta^3 p_{n_s-3,h}^s & (i = m-3),\\ 0 & (0 \le i \le 3;\ i = m),\end{cases} \tag{7.6}$$

$$\frac{\partial r_{ih}}{\partial \mu_2} = \begin{cases} \frac{n_s}{(m-1)_2}(1 - t_{s-1})^{-1} \Delta p_{n_s-1,h}^s & (i = m-2),\\ 3\frac{n_s}{(m-1)_2}(1 - t_{s-1})^{-1} \Delta p_{n_s-1,h}^s - 3\mu_1 \frac{(n_s-1)_2}{(m-2)_3}(1 - t_{s-1})^{-2} \Delta^2 p_{n_s-2,h}^s & (i = m-3),\\ 0 & (0 \le i \le 3;\ i = m-1, m),\end{cases} \tag{7.7}$$

$$\frac{\partial r_{ih}}{\partial \mu_3} = \begin{cases} -\frac{n_s}{(m-2)_3}(1-t_{s-1})^{-1}\Delta p^s_{n_s-1,h} & (i = m-3), \\ 0 & (0 \le i \le 3;\ m-2 \le i \le m). \end{cases} \tag{7.8}$$

(cf. (4.11)–(4.16)).
Notice that the partial derivatives of hth coordinates of the inner control points (6.7) depend on (7.3)–(7.8) in the following way:

$$\frac{\partial r_{ih}}{\partial \lambda_u} = -\frac{1}{2m+1}\sum_{j=k+1}^{m-l-1}\binom{m}{j}c_{ji}\sum_{g=u}^{k}\binom{2m}{j+g}^{-1}\binom{m}{g}\frac{\partial r_{gh}}{\partial \lambda_u}, \tag{7.9}$$

$$\frac{\partial r_{ih}}{\partial \mu_v} = -\frac{1}{2m+1}\sum_{j=k+1}^{m-l-1}\binom{m}{j}c_{ji}\sum_{g=m-l}^{m-v}\binom{2m}{j+g}^{-1}\binom{m}{g}\frac{\partial r_{gh}}{\partial \mu_v} \tag{7.10}$$

with c_{ji} being introduced in Lemma 3.14. When k, $l \le 3$, we compute $\frac{\partial r_{ih}}{\partial \lambda_u}$, $\frac{\partial r_{ih}}{\partial \mu_v}$ by (7.9), (7.10) if $k < i < m-l$, and by (7.3)–(7.8) otherwise. Finally, we put the expressions (7.3)–(7.10) into the equations of the system (7.2).

As in the case of the analogical degree reduction problem, the obtained system in nonlinear for $k \ge 2$ or $l \ge 2$. Moreover, we must guarantee that λ_1, $\mu_1 > 0$, which results in the same directions of tangent vectors at the endpoints of the curves (1.25) and (1.26) (cf. Remark 4.3). Thus, to solve Problem 1.11 in its most general form, we must solve the quadratic or nonlinear programming problem of minimizing (7.1) subject to

$$\lambda_1 \ge z_0, \quad \mu_1 \ge z_1, \tag{7.11}$$

where z_0 and z_1 are prescribed positive lower bounds. The idea behind this approach is basically the same as in §4.2.1. Hence, we omit further explanation. For the algorithm, see §7.2.1.

Another approach, which works relatively well for both degree reduction and merging, is the $C^{p,q}/G^{k,l}$ simplification (see §2.3). In the cases of $k = l = 2$ and $k = l = 3$, this idea was first presented in [69, 71]. As in §4.2.2, it can be generalized for any k and l not exceeding 3. Since the rules of simplification are the same, we assume that no further explanation is required. An outline of the algorithm is given in §7.2.2.

7.2 Algorithms

In this subsection, we provide some details of algorithmic implementation of the result given in Theorem 6.6 combined with the methods of computing the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$ (see §7.1).

7.2.1 $G^{k,l}$-constrained merging algorithm

Now, we give two-phase algorithm of solving Problem 1.11 in its most general form. See Algorithm 7.1 (cf. Algorithm 4.9).

During Phase A, the task is to minimize the error function (7.1) subject to the constraints (7.11). After we build the error function, we solve the constrained quadratic or nonlinear programming problem and obtain optimal values for the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$. Phase B of the algorithm is to compute the control points of the curve (1.26). Using

Theorem 6.6, we determine them with the complexity $O(sm^2)$, which should be compared to the complexity $O(sm^3)$ of other known methods (see, e.g., [69, 71, 72, 75]). Moreover, in contrast to those methods, Algorithm 7.1 works for any k, l such that $-1 \leq k, l \leq 3$. According to the author's knowledge, this is the fastest and the most general algorithm available.

Notice that when $k, l < 1$, Problem 1.11 is the same as Problem 1.10. Therefore, to obtain a proper result, one can just execute Algorithm 6.8.

Algorithm 7.1. *[$G^{k,l}$-constrained merging of Bézier curves]*

Input: n_i $(i = 1, 2, \ldots, s)$ – *degrees of the curves* (1.25);
p_j^i $(j = 0, 1, \ldots, n_i;\ i = 1, 2, \ldots, s)$ – *control points of the curves* (1.25);
m – *degree of the curve* (1.26);
k, l – *orders of the geometric continuity (see* (1.27)*);*
$0 = t_0 < t_1 < \ldots < t_s = 1$ – *partition of the interval* [0, 1];
z_0, z_1 – *lower bounds for the parameters* λ_1 *and* μ_1*, respectively (see* (7.11)*)*

Assumptions: $m \geq \max_i n_i$; $z_0, z_1 > 0$; $k \leq n_1$; $l \leq n_s$; $-1 \leq k, l \leq 3$; $k + l < m - 1$

Output: *control points of the $G^{k,l}$-constrained merged Bézier curve*

Phase A

Step 1. *Check if the considered case can be solved using Algorithm 6.8*

If *($k, l < 1$)* ***then*** *execute Algorithm 6.8, and return* $r_0, r_1, \ldots, r_m$.

Step 2. *Compute* $c_{ij}(m, k, l)$ $(i, j = k + 1, k + 2, \ldots, m - l - 1)$ *using Algorithm 3.15.*

Step 3. *Compute* $d_{jh}^{(i)}$ $(i = 1, 2, \ldots, s;\ j, h = 0, 1, \ldots, m)$ *using Algorithm 6.4.*

Step 4. *Compute* $E(\lambda_1, \lambda_2, \ldots, \lambda_k, \mu_1, \mu_2, \ldots, \mu_l)$ *by* (7.1).

Step 5. *Determine set c of constraints*

(i) $c := \{\lambda_1 \geq z_0,\ \mu_1 \geq z_1\}$;

(ii) ***If*** *($k < 1$)* ***then*** $c := c \setminus \{\lambda_1 \geq z_0\}$;

(iii) ***If*** *($l < 1$)* ***then*** $c := c \setminus \{\mu_1 \geq z_1\}$.

Step 6.

If *($k > 1$ or $l > 1$)* ***then***

- *obtain* $\lambda_1, \lambda_2, \ldots, \lambda_k$, *and* $\mu_1, \mu_2, \ldots, \mu_l$ *by solving the nonlinear programming problem of minimizing the error* (7.1) *subject to the constraints c;*

else

- *obtain* $\lambda_1, \lambda_2, \ldots, \lambda_k$, *and* $\mu_1, \mu_2, \ldots, \mu_l$ *by solving the quadratic programming problem of minimizing the error* (7.1) *subject to the constraints c.*

Phase B

Step 7. *Compute* $r_0, r_1, \ldots, r_k$ *by* (2.17)–(2.19).

Step 8. *Compute* $r_m, r_{m-1}, \ldots, r_{m-l}$ *by* (2.20)–(2.22).

`Step 9.` *Compute* $\hat{p}_j^i$ $(i = 1, 2, \ldots, s;\ j = 0, 1, \ldots, m)$ *by* (6.9).

`Step 10.` *Compute* $\hat{r}_j$ $(j = k+1, k+2, \ldots, m-l-1)$ *by* (6.8).

`Step 11.` *Compute* r_j $(j = k+1, k+2, \ldots, m-l-1)$ *by* (6.7).

`Step 12.` *Return* $r_0, r_1, \ldots, r_m$.

7.2.2 Outline of the $C^{p,q}/G^{k,l}$-constrained merging algorithm

Now, we give the outline of two-phase algorithm of $C^{p,q}/G^{k,l}$-constrained merging of Bézier curves. The method is analogical to that of $C^{p,q}/G^{k,l}$-constrained degree reduction of Bézier curves (see Algorithm 4.8). Because of this similarity, we omit a detailed description of the full algorithm. Moreover, it is also possible to derive explicit formulas for the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$ in some of the selected cases (cf. Remark 4.4 and §4.3). Therefore, one could give an algorithm which would be similar to Algorithm 4.7. However, due to extensive mathematical details associated with this approach, it is omitted in the thesis.

The simplifying assumptions that $\lambda_1 := 1$ when $k > 1$, and $\mu_1 := 1$ when $l > 1$, make the system (7.2) linear. Consequently, in Phase A, the algorithm solves this linear system to compute the continuity parameters $\{\lambda_i\}$ and $\{\mu_j\}$. Note that when $k = 1$ or $l = 1$, a solution of the system may not satisfy $\lambda_1 > 0$ or $\mu_1 > 0$, respectively. In such cases, we must deal with the quadratic programming problem subject to the conditions (7.11). Phase B is the same as for Algorithm 7.1.

Apart from very rare cases which force the algorithm to solve the quadratic programming problem, the computational cost is lower than that of Algorithm 7.1. On the other hand, because of the simplification, the results are less accurate (see §7.3).

7.3 Examples

In this subsection, we show the effectiveness of the methods. Taking into account the different types of continuity constraints, we compare the following cases:

(i) $C^{k,l}$-constrained case, solved using Algorithm 6.8;

(ii) $C^{p,q}/G^{k,l}$-constrained case, solved using the algorithm described in §7.2.2;

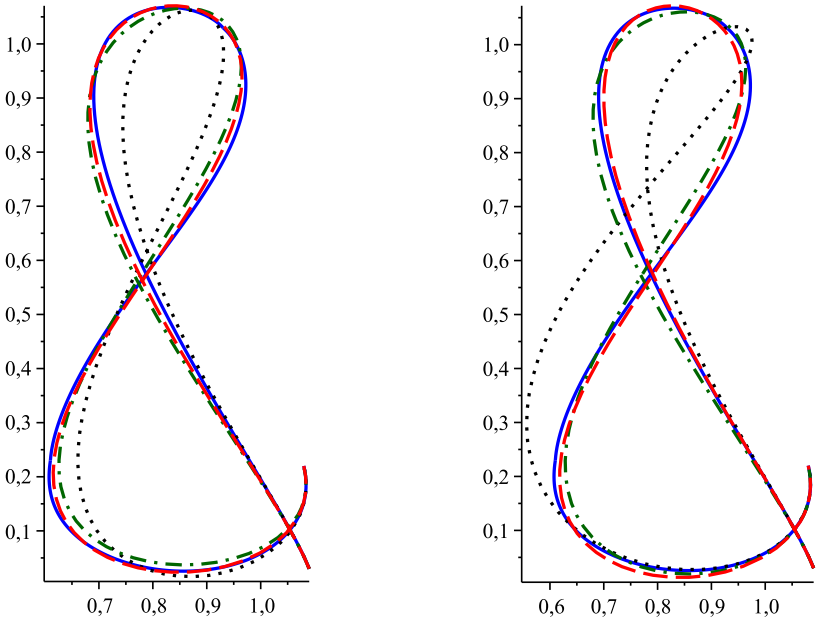
(iii) $G^{k,l}$-constrained case, solved using Algorithm 7.1.

In each case, we give L_2-error $E_2 := \sqrt{E}$ (see (7.1)) and maximum error E_∞ (1.11).

The results have been obtained in Maple™13 using 32-digit arithmetic. As in §4.5, we use Maple™ `fsolve` procedure, in the $C^{p,q}/G^{k,l}$-constrained case, to solve the system of linear equations, and `QPSolve`, `NLPSolve` procedures, to deal with the quadratic and nonlinear programming problems, respectively. Initial points for both procedures correspond to the values of continuity parameters in the $C^{k,l}$-constrained case (see Remark 2.3).

Example 7.2. First, we recall the composite Bézier curve "Ampersand". For the control points and the partition of the unit interval, see Example 6.10. We look for Bézier curves of degrees 7 and 8 satisfying the different types of continuity constraints. For the results, see Table 12 and Figure 21. Clearly, the $C^{k,l}$ constraints lead to by far the worst results,

which are unacceptable from a practical point of view. The difference between the $C^{p,q}/G^{k,l}$-constrained and $G^{k,l}$-constrained cases is noticeable and becomes significant for larger values of k and l.



(a) (b)

Figure 21: Merging of three segments of the composite Bézier curve. The original composite curve (blue solid line) and the merged curves with $C^{k,l}$ (black dotted line), $C^{p,q}/G^{k,l}$ (green dash-dotted line) and $G^{k,l}$ (red dashed line) continuity constraints; parameters: (a) $m = 8$, $k = 3$, $l = 2$; (b) $m = 8$, $k = l = 3$.

Parameters			$C^{k,l}$ solution		$C^{p,q}/G^{k,l}$ solution		$G^{k,l}$ solution	
m	k	l	E_2	E_∞	E_2	E_∞	E_2	E_∞
7	2	2	$6.30e{-2}$	$1.18e{-1}$	$5.29e{-2}$	$9.92e{-2}$	$1.83e{-2}$	$3.83e{-2}$
7	2	3	$2.08e{-1}$	$3.54e{-1}$	$5.36e{-2}$	$1.01e{-1}$	$3.02e{-2}$	$4.87e{-2}$
7	3	2	$9.10e{-2}$	$1.85e{-1}$	$5.62e{-2}$	$1.04e{-1}$	$1.90e{-2}$	$4.12e{-2}$
8	2	2	$1.59e{-2}$	$2.88e{-2}$	$1.39e{-2}$	$2.48e{-2}$	$7.69e{-3}$	$1.34e{-2}$
8	2	3	$2.01e{-2}$	$4.02e{-2}$	$1.40e{-2}$	$2.44e{-2}$	$1.05e{-2}$	$2.17e{-2}$
8	3	2	$3.86e{-2}$	$6.42e{-2}$	$1.70e{-2}$	$2.98e{-2}$	$8.48e{-3}$	$1.32e{-2}$
8	3	3	$7.21e{-2}$	$1.33e{-1}$	$1.87e{-2}$	$3.23e{-2}$	$1.35e{-2}$	$2.55e{-2}$

Table 12: L_2-errors and maximum errors of constrained merging of three segments of the composite Bézier curve "Ampersand" with the different types of continuity constraints.

Example 7.3. Secondly, we present the composite Bézier curve "H", formed by four cubic segments which are defined by the control points $\{(0.39, 1.24), (0.52, 1.08), (0.57, 1.52), (0.6, 1.02)\}$, $\{(0.6, 1.02), (0.63, 0.7), (0.49, 0.24), (0.38, 0.6)\}$, $\{(0.38, 0.6), (0.33, 0.84), (1.38, 1), (1.13, 1.25)\}$ and $\{(1.13, 1.25), (0.8, 1.5), (1.01, 0.15), (1.12, 0.55)\}$, respectively. According to (6.15), we have $t_0 = 0$, $t_1 \doteq 0.14$, $t_2 \doteq 0.36$, $t_3 \doteq 0.69$, $t_4 = 1$. As we consider the different types of continuity constraints, it can be seen that, once again, the result of the $C^{k,l}$-constrained case is unsatisfactory. See Figure 22.

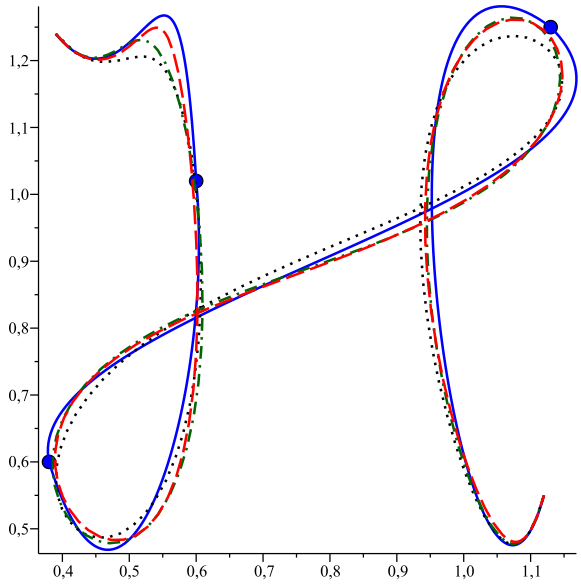


Figure 22: Merging of four segments of the composite Bézier curve. The original composite curve (blue solid line) and the merged curves of degree 11 with $C^{2,3}$ (black dotted line; errors: $E_2 = 2.94e{-}2$ and $E_\infty = 7.90e{-}2$), $C^{1,1}/G^{2,3}$ (green dash-dotted line; errors: $E_2 = 1.97e{-}2$ and $E_\infty = 5.54e{-}2$) and $G^{2,3}$ (red dashed line; errors: $E_2 = 1.75e{-}2$ and $E_\infty = 5.80e{-}2$) continuity constraints.

Chapter 8

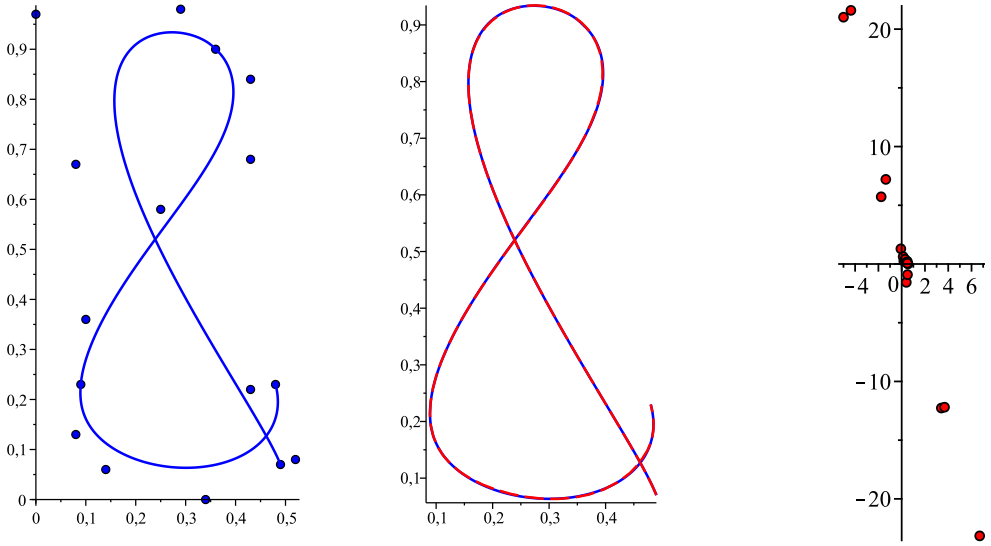
Merging of planar Bézier curves with box constraints

In this chapter, we propose a new approach to the problem of $C^{k,l}$-constrained merging of planar Bézier curves with respect to the L_2-norm. Recall that in Chapter 5, we observe that as a result of the traditional degree reduction of planar Bézier curves (see Problem 5.1), computed control points can be located far away from the plot of the curve. We also explain why this is a serious defect (see §5.1). Next, to eliminate that issue, we solve the problem of degree reduction with box constraints (see Problem 5.5). In this chapter, we show that the same observations may apply to control points of the merged curve. An illustrative example of such a situation is presented in §8.1. The problem of merging of planar Bézier curves with box constraints is formulated in §8.2, and its solution is given in §8.3. Some other examples are shown in §8.4. This chapter is based on paper [45].

8.1 Motivation

As it turns out, the observations on the degree reduction problem (see §5.1) also apply to the problem of merging. In order to see the issue clearly, let us consider the following example.

Example 8.1. Once again, recall the composite Bézier curve "Ampersand" which consists of three 5th degree Bézier segments. Here the control points are translated and scaled; we have $\{(0.49, 0.07), (0.43, 0.22), (0.08, 0.67), (0, 0.97), (0.29, 0.98), (0.36, 0.9)\}$, $\{(0.36, 0.9), (0.43, 0.84), (0.43, 0.68), (0.25, 0.58), (0.1, 0.36), (0.09, 0.23)\}$, and $\{(0.09, 0.23), (0.08, 0.13), (0.14, 0.06), (0.34, 0), (0.52, 0.08), (0.48, 0.23)\}$. For the partition of the unit interval, see Example 6.10. Now, consider a single Bézier curve of degree 16 being the solution of Problem 1.10 for $k = 2$ and $l = 0$. Figure 23b shows both curves. Clearly, the result of the approximation is very accurate; errors: $E = 8.01e{-}4$ (see (1.23)) and $E_\infty = 3.03e{-}3$ (see (1.11)). Observe also that the original control points are quite close to the plot of the curve (see Figure 23a). In contrast, the resulting control points are located far away from the plot of the curve (see Figure 23c). Note that we are unable to see the curve and its control points in one figure. Because of the non-intuitive location of the control points, further modeling of the merged curve is hard to imagine. Notice also that the resulting convex hull is huge, therefore, completely useless. See Remarks 5.2 and 5.3. Comparing this result with the ones from Chapter 5, we see that the defect seems to be even more significant (cf. Figures 11b, 14b, 15d and 17c).



(a) (b) (c)

Figure 23: Figure (a) shows the original composite Bézier curve with its control points. Figure (b) illustrates the original composite Bézier curve (blue solid line) and the $C^{2,0}$-constrained merged Bézier curve (red dashed line) being the solution of Problem 1.10. Figure (c) presents the control points of the merged curve (red color).

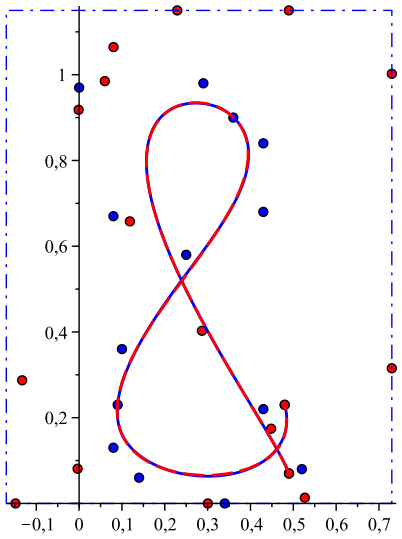


Figure 24: The original composite Bézier curve (blue solid line with blue control points) and the $C^{2,0}$-constrained merged Bézier curve (red dashed line with red and green control points) satisfying certain box constraints (blue dotted-dashed frame). The control points which are constrained by the continuity conditions are green, while the other ones are red and restricted by the box constraints.

Now, let us impose some box constraints. We want to place the searched control points inside the specified rectangular area (including edges of the rectangle). The resulting curve is illustrated in Figure 24; parameters: $m = 16$, $k = 2$, $l = 0$. Notice that the approximation is quite accurate; errors: $E = 1.41e{-}3$ and $E_\infty = 3.72e{-}03$. Moreover, in this case, the computed control points are located much closer to the merged curve. What is more, we have obtained much smaller convex hull, which can be used to solve efficiently some important problems (see Remark 5.3). The resulting curve is a solution of the new problem of merging which we formulate in the next subsection. More examples can be found in §8.4.

8.2 Problem of merging of planar Bézier curves with box constraints

In this subsection, we formulate the following new problem of merging of planar Bézier curves (cf. Problems 1.10 and 5.5).

Problem 8.2. *[Merging of planar Bézier curves with box constraints]*
Let $0 = t_0 < t_1 < \ldots < t_s = 1$ be a partition of the interval $[0, 1]$. Let there be given a composite Bézier curve $P(t) = [P_x(t), P_y(t)]$ $(t \in [0, 1])$ which in the interval $[t_{i-1}, t_i]$ $(i = 1, 2, \ldots, s)$ is exactly represented as a Bézier curve $P^i(t) = \left[P^i_x(t), P^i_y(t)\right] \in \Pi^2_{n_i}$, i.e.,

$$P(t) = P^i(t) := \sum_{j=0}^{n_i} p^i_j \, B^{n_i}_j(u_i(t)) \equiv \mathbf{b}_{n_i,u_i(t)}\mathbf{p}^i \qquad (t \in [t_{i-1},\, t_i]),$$

where $u_i(t) := \frac{t-t_{i-1}}{\Delta t_{i-1}}$, $\mathbf{b}_{n,t} := [B^n_0(t), B^n_1(t), \ldots, B^n_n(t)]$, and $\mathbf{p}^i := \left[p^i_0, p^i_1, \ldots, p^i_{n_i}\right]^T$ with $p^i_j := (p^{i,x}_j, p^{i,y}_j) \in \mathbb{R}^2$. Find a Bézier curve

$$R(t) := \sum_{j=0}^{m} r_j \, B^m_j(t) \equiv \mathbf{b}_{m,t}\mathbf{r} \qquad (t \in [0,\, 1]),$$

where $R(t) = [R_x(t), R_y(t)] \in \Pi^2_m$, and $\mathbf{r} := [r_0, r_1, \ldots, r_m]^T$ with $r_i := (r^x_i, r^y_i) \in \mathbb{R}^2$, satisfying the following conditions:

(i) L_2-error

$$E := \|P - R\|_{L_2} = \sqrt{\int_0^1 \|P(t) - R(t)\|^2 \, \mathrm{d}t} \tag{8.1}$$

is minimized in the space Π^2_m;

(ii) P and R are $C^{k,l}$-continuous at the endpoints, i.e.,

$$\left.\begin{aligned} P^{(i)}(0) &= R^{(i)}(0) \qquad (i = 0, 1, \ldots, k), \\ P^{(j)}(1) &= R^{(j)}(1) \qquad (j = 0, 1, \ldots, l), \end{aligned}\right\} \tag{8.2}$$

where $k \le n_1$, $l \le n_s$, $k, l \ge -1$ and $k + l < m - 1$;

(iii) control points r_i $(k < i < m - l)$ *are located inside the specified rectangular area, including edges of the rectangle, i.e., the following box constraints are fulfilled:*

$$c_z \le r_i^z \le C_z \qquad (i = k+1, k+2, \ldots, m-l-1;\ z = x, y), \tag{8.3}$$

where $c_x, c_y, C_x, C_y \in \mathbb{R}$.

Remark 8.3. In Chapter 6, we are dealing with the *traditional merging*, i.e., the minimization of (8.1), with the conditions (8.2), but without the box constraints (8.3) (see Problem 1.10). In addition, a reasonable assumption that $m \ge \max_i n_i$ is made.

8.3 Solution

Now, we give the method of solving Problem 8.2.

First, we notice that some of the observations concerning Problem 5.5 are also true in the case of Problem 8.2. As in §5.2, a planar Bézier curve being the solution of Problem 8.2 can be obtained in a componentwise way. Therefore, it is sufficient to deal with the case of $z = x$ coordinate (cf. (8.3)). Moreover, observe that this property can be generalized. As a result, the method given in this thesis can be easily applied to three dimensional Bézier curves.

Next, we recall that the conditions (8.2) yield the well-known formulas (2.5) and (2.6) for the control points $r_0, r_1, \ldots, r_k$ and $r_m, r_{m-1}, \ldots, r_{m-l}$, respectively.

What remains is to minimize

$$E_x(\mathbf{r}_x) := \sqrt{\int_0^1 (P_x(t) - R_x(t))^2 \,\mathrm{d}t},$$

where $\mathbf{r}_x := [r_0^x, r_1^x, \ldots, r_m^x]^T$, subject to the conditions (8.3) for $z = x$. One can easily see that $E_x^2(\mathbf{r}_x)$ is a quadratic function. Therefore, in the following subsection, we are dealing with the box-constrained quadratic programming problem.

8.3.1 Quadratic programming with box constraints

In this subsection, we use the quadratic programming approach to solve Problem 8.2.

Recall that in §5.3, we mention the following quadratic programming problem with box constraints:

$$\min_{\mathbf{v} \le \mathbf{x} \le \mathbf{w}} \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}, \tag{8.4}$$

where $\mathbf{x}, \mathbf{v}, \mathbf{w}, \mathbf{c} \in \mathbb{R}^i$ and $\mathbf{Q} \in \mathbb{R}^{i \times i}$.

Now, we set $\mathbf{v} := [c_x, c_x, \ldots, c_x]^T \in \mathbb{R}^{m-k-l-1}$, $\mathbf{w} := [C_x, C_x, \ldots, C_x]^T \in \mathbb{R}^{m-k-l-1}$ and $\mathbf{x} := \mathbf{r}_x^{\mathcal{F}}$, where we define $\mathcal{F} := \{k+1, k+2, \ldots, m-l-1\}$ and use the notation of (5.8). Next, we should adjust $E_x^2(\mathbf{r}_x)$ to the form (8.4).

Taking into account that P_x is a piecewise polynomial, we have to subdivide the searched polynomial R_x as well. Note that this can be done by applying de Casteljau algorithm (see §1.4, pt. 9). According to Lemmas 6.1 and 6.3, we have

$$R_x(t) = R_x^i(t) := \sum_{j=0}^{m} r_j^{i,x} B_j^m(u_i(t)) \equiv \mathbf{b}_{m,u_i(t)}\mathbf{D}_i\mathbf{r}_x \qquad (t \in [t_{i-1},\, t_i];\ i = 1, 2, \ldots, s),$$

where $\mathbf{D}_i = \left[d_{jh}^{(i)}\right] \in \mathbb{R}^{(m+1)\times(m+1)}$. Recall that the entries $d_{jh}^{(i)}$ $(i = 1, 2, \ldots, s;\ j, h = 0, 1, \ldots, m)$ can be computed with the complexity $O(sm^2)$ using Algorithm 6.4.

Remark 8.4. The efficient approach presented in §6.1 should be compared with the approach of Lu [72, §2], where

$$\mathbf{D}_i := \mathbf{A}_1(t_{i-1}/t_i)\mathbf{A}_2(t_i) \qquad (i = 1, 2, \ldots, s) \tag{8.5}$$

with

$$\mathbf{A}_1(\lambda) = \begin{bmatrix} B_0^m(\lambda) & B_1^m(\lambda) & \cdots & B_m^m(\lambda) \\ 0 & B_0^{m-1}(\lambda) & \cdots & B_{m-1}^{m-1}(\lambda) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \quad \mathbf{A}_2(\lambda) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ B_0^1(\lambda) & B_1^1(\lambda) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_0^m(\lambda) & B_1^m(\lambda) & \cdots & B_m^m(\lambda) \end{bmatrix}.$$

Observe that the direct use of (8.5) results in the complexity $O(sm^3)$.

Next, assuming that $\mathcal{K} := \{0, 1, \ldots, m\}$, $\mathcal{C} := \mathcal{K} \setminus \mathcal{F}$ and using the notation of (5.8), we write (cf. [72, (11)])

$$\begin{aligned} E_x^2(\mathbf{r}_x) &= \int_0^1 (P_x(t) - R_x(t))^2 \,\mathrm{d}t = \sum_{i=1}^{s} \int_{t_{i-1}}^{t_i} \left(P_x^i(t) - R_x^i(t)\right)^2 \,\mathrm{d}t \\ &= \sum_{i=1}^{s} \Delta t_{i-1} \int_0^1 \left(\mathbf{b}_{n_i,v}\mathbf{p}^{i,x} - \mathbf{b}_{m,v}\mathbf{D}_i^{\mathcal{K},\mathcal{C}}\mathbf{r}_x^{\mathcal{C}} - \mathbf{b}_{m,v}\mathbf{D}_i^{\mathcal{K},\mathcal{F}}\mathbf{r}_x^{\mathcal{F}}\right)^2 \,\mathrm{d}v \\ &= \mathbf{c}^T\mathbf{r}_x^{\mathcal{F}} + \frac{1}{2}\left(\mathbf{r}_x^{\mathcal{F}}\right)^T \mathbf{Q}\mathbf{r}_x^{\mathcal{F}} + a \;=:\; g\left(\mathbf{r}_x^{\mathcal{F}}\right) + a, \end{aligned}$$

where

$$\mathbf{p}^{i,x} := \left[p_0^{i,x}, p_1^{i,x}, \ldots, p_{n_i}^{i,x}\right]^T, \qquad \mathbf{Q} := 2\sum_{i=1}^{s} \Delta t_{i-1} \left(\mathbf{D}_i^{\mathcal{K},\mathcal{F}}\right)^T \mathbf{G}_{m,m}\mathbf{D}_i^{\mathcal{K},\mathcal{F}},$$

$$\mathbf{c} := 2\sum_{i=1}^{s} \Delta t_{i-1} \left(\mathbf{D}_i^{\mathcal{K},\mathcal{F}}\right)^T \left(\mathbf{G}_{m,m}\mathbf{D}_i^{\mathcal{K},\mathcal{C}}\mathbf{r}_x^{\mathcal{C}} - \mathbf{G}_{m,n_i}\mathbf{p}^{i,x}\right),$$

and $a \in \mathbb{R}$ is a certain constant term. Here $\mathbf{G}_{M,N} := \left[g_{ij}^{M,N}\right] \in \mathbb{R}^{(M+1)\times(N+1)}$ is the well-known *Gramian matrix of the Bernstein basis* with the elements

$$g_{ij}^{M,N} := \frac{1}{M+N+1}\binom{M}{i}\binom{N}{j}\binom{M+N}{i+j}^{-1} \qquad (i = 0, 1, \ldots, M;\ j = 0, 1, \ldots, N).$$

Obviously, a is meaningless in the minimization process, therefore, the *significant terms* of $E_x^2(\mathbf{r}_x)$ are given by $g(\mathbf{r}_x^{\mathcal{F}})$, which is written in the form (8.4).

Remark 8.5. Matrix $\mathbf{Q}$ is positive definite (see [72, §3.1]), therefore, the objective function g is strictly convex. Furthermore, the feasible set is nonempty, closed and convex. We conclude that the quadratic programming problem has a unique solution (see, e.g., [29, Proposition 2.5]) and so does Problem 8.2. In contrast, a solution of the analogical degree reduction problem may not be unique (cf. Theorem 5.6). The difference is that, in this chapter, we consider the *continuous inner product* (see (8.1)) instead of the *discrete inner product* (see (5.3)).

Recall that there are many papers dealing with the box-constrained quadratic programming problem. Some of them are mentioned in §5.3.

8.4 Examples

In this subsection, we show several examples of application of the discussed method. A solution of Problem 1.10 is computed using Algorithm 6.8. To solve the quadratic programming problem with box constraints (8.4), we use the matrix version of Maple$^{\text{TM}}$ `QPSolve` command. It is worth noting that this procedure implements an iterative active set method and it is suited for the box constraints, i.e., the vectors of lower and upper bounds can be passed using the optional parameter `bd`. According to the documentation provided by Maplesoft$^{\text{TM}}$, in the case of the convex optimization, a global minimum is returned (cf. Remark 8.5). For the initial point, we choose the lower bounds, i.e., c_x and c_y. The results have been obtained on a computer with `Intel Core i5-3337U 1.8GHz` processor and `8GB` of `RAM`, using Maple$^{\text{TM}}$13 with 24-digit arithmetic.

Example 8.6. We recall the composite Bézier curve "D" (see Figure 25a) formed by three cubic segments. Here the control points are translated and scaled; we have $\{(0.32, 0.81), (0.26, 0.59), (0.18, 0), (0.06, 0.27)\}$, $\{(0.06, 0.27), (0, 0.42), (0.42, 0.08), (0.57, 0.25)\}$ and $\{(0.57, 0.25), (0.76, 0.46), (0.8, 1), (0.22, 0.85)\}$. For the partition of the unit interval, see Example 1.3. Figure 25b shows the result of the traditional $C^{k,l}$-constrained merging for $m = 18$, $k = 0$, $l = 1$. The merged curve looks like a perfect approximation (errors: $E = 3.25e{-}3$ and $E_\infty = 9.67e{-}3$), unfortunately, it suffers from the defect described in §8.1 (see Figure 25c). To avoid this, we solve Problem 8.2 for $m = 18$, $k = 0$, $l = 1$, with the following box constraints:

$$\begin{aligned} c_x &:= \min_{1\le i\le s}\min_{0\le j\le n_i} p_j^{i,x} - 0.2 = -0.2, & C_x &:= \max_{1\le i\le s}\max_{0\le j\le n_i} p_j^{i,x} = 0.8, \\ c_y &:= \min_{1\le i\le s}\min_{0\le j\le n_i} p_j^{i,y} - 0.3 = -0.3, & C_y &:= \max_{1\le i\le s}\max_{0\le j\le n_i} p_j^{i,y} = 1 \end{aligned} \tag{8.6}$$

(cf. (8.3)), and obtain the curve shown in Figure 25d (errors: $E = 1.28e{-}2$ and $E_\infty = 3.01e{-}2$). Compare Figure 25d with Figure 25c to see a big difference in the location of the resulting control points. Obviously, the curve in Figure 25d is much more satisfying in this regard.

Example 8.7. Now, we consider the composite Bézier curve with four 5th degree Bézier segments (see Figure 26a). For the original control points, see [72, Example 3]. Here each coordinate of the control points is divided by 5.1. According to (6.15), we get $t_0 = 0$, $t_1 \doteq 0.24$, $t_2 \doteq 0.49$, $t_3 \doteq 0.76$, $t_4 = 1$. As a result of the traditional $C^{k,l}$-constrained merging ($m = 19$, $k = l = 0$), we obtain the Bézier curve which is illustrated in Figure 26b. Once again, we get a good approximation (errors: $E = 2.08e{-}3$ and $E_\infty = 5.65e{-}3$), however, the resulting control points are located far away from the plot of the curve (see Figure 26c). Taking into account the axis scale in Figure 26c, we conclude that this example seems to be extremely difficult. Nonetheless, the solution of Problem 8.2 for $m = 19$, $k = l = 0$, with the box constraints

$$\begin{aligned} c_x &:= \min_{1\le i\le s}\min_{0\le j\le n_i} p_j^{i,x} - 0.2 = -0.2, & C_x &:= \max_{1\le i\le s}\max_{0\le j\le n_i} p_j^{i,x} + 0.2 \doteq 0.65, \\ c_y &:= \min_{1\le i\le s}\min_{0\le j\le n_i} p_j^{i,y} - 0.2 = -0.2, & C_y &:= \max_{1\le i\le s}\max_{0\le j\le n_i} p_j^{i,y} + 0.2 = 1.2 \end{aligned}$$

is quite decent (errors: $E = 9.71e{-}3$ and $E_\infty = 1.90e{-}2$). See Figure 26d.

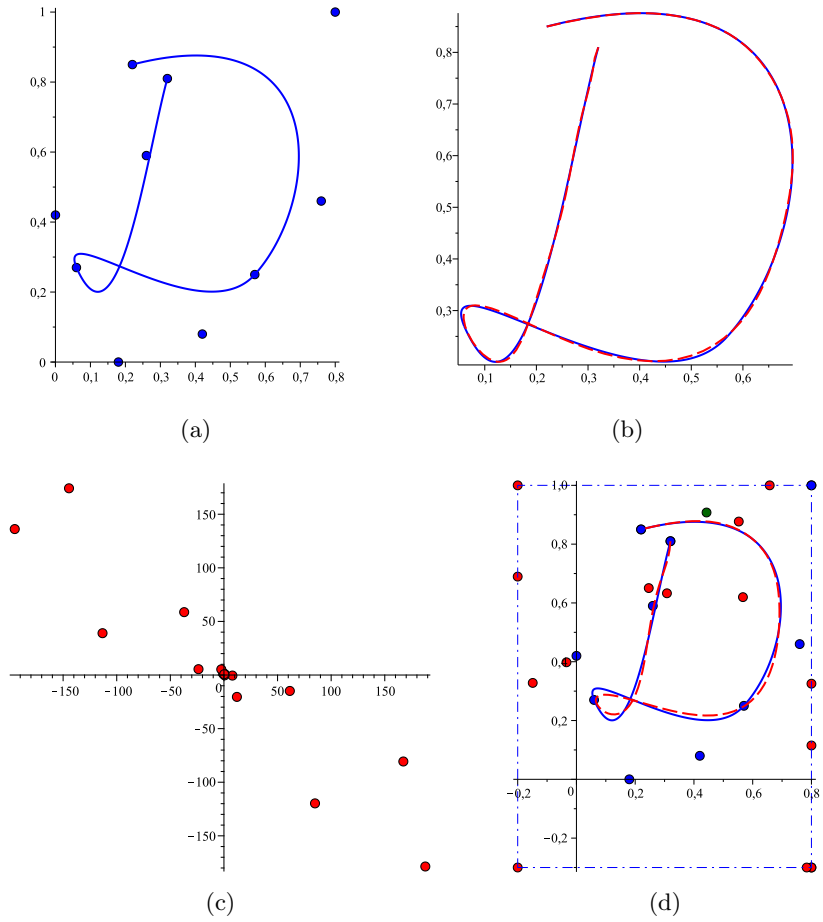


Figure 25: Merging of three segments of the composite Bézier curve. The original composite curve (blue solid line with blue control points) and the merged curve (red dashed line with red and green control points); parameters: $m = 18$, $k = 0$, $l = 1$. Figure (a) shows the original composite curve with its control points. Figure (b) illustrates the curve being the solution of Problem 1.10. Figure (c) presents the control points of the merged curve shown in Figure (b). The curve being the solution of Problem 8.2 with its control points is shown in Figure (d). The control points which are constrained by the continuity conditions are green, while the other ones are red and in the case of (d) bounded by the blue dotted-dashed frame.

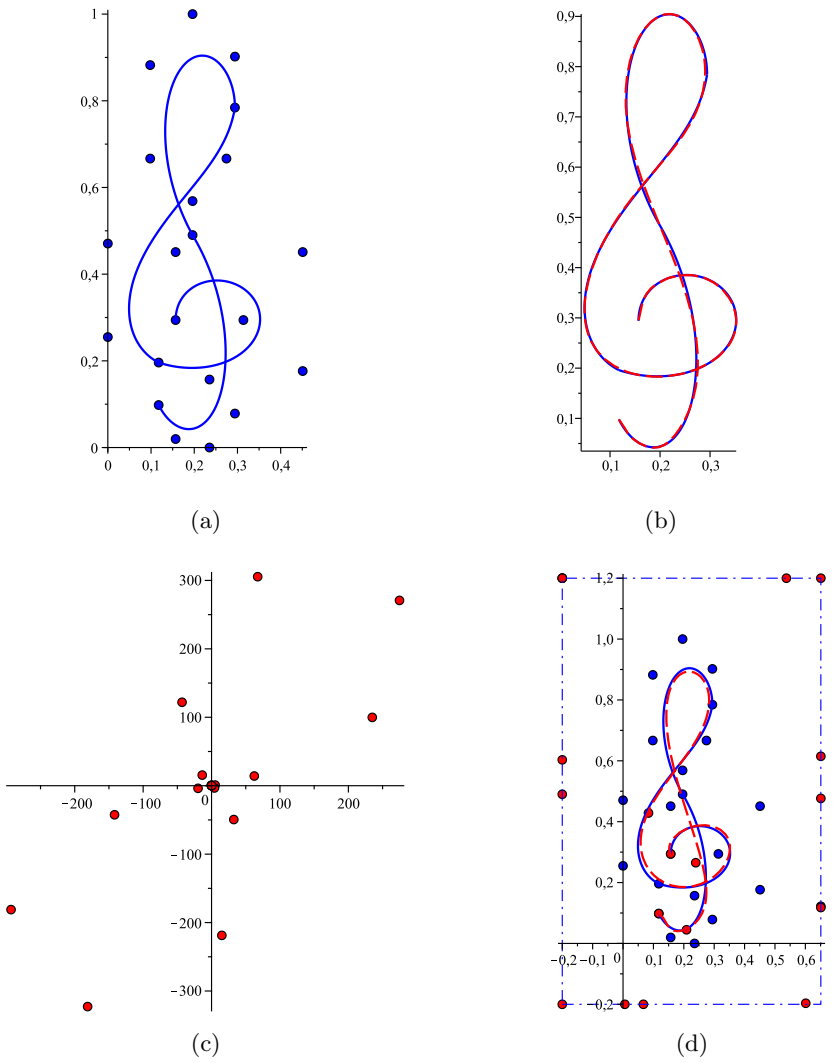


Figure 26: Merging of four segments of the composite Bézier curve. The original composite curve (blue solid line with blue control points) and the merged curve (red dashed line with red control points); parameters: $m = 19$, $k = l = 0$. Figure (a) shows the original composite curve with its control points. Figure (b) illustrates the curve being the solution of Problem 1.10. Figure (c) presents the control points of the merged curve shown in Figure (b). The curve being the solution of Problem 8.2 with its control points and the rectangular area (blue dotted-dashed frame) are shown in Figure (d).

Remark 8.8. As stated in Remark 5.15, selection of the rectangular area is a difficult issue. The choice always depends on the considered example and on the precision level that we accept as satisfactory. However, there is a strategy that seems to work quite well for the given examples. To explain this procedure, let us revisit Example 8.6. At the beginning, we set

$$\begin{aligned} c_x^{(1)} &:= \min_{1\le i\le s}\min_{0\le j\le n_i} p_j^{i,x} = 0, & C_x^{(1)} &:= \max_{1\le i\le s}\max_{0\le j\le n_i} p_j^{i,x} = 0.8, \\ c_y^{(1)} &:= \min_{1\le i\le s}\min_{0\le j\le n_i} p_j^{i,y} = 0, & C_y^{(1)} &:= \max_{1\le i\le s}\max_{0\le j\le n_i} p_j^{i,y} = 1. \end{aligned} \tag{8.7}$$

Consequently, the resulting control points will be bounded by the outermost control points

of the original curves. Unfortunately, the obtained curve is unsatisfactory (see Figure 27a). Next, to improve this result, we must expand the rectangular area. Intuition tells us that we should try to move the borders with the highest numbers of the control points. We consider

$$\begin{aligned} c_x^{(2)} &:= c_x^{(1)} - 0.04w_1 \doteq -0.05, \qquad & C_x^{(2)} &:= C_x^{(1)}, \\ c_y^{(2)} &:= c_y^{(1)} - 0.04w_1 \doteq -0.05, \qquad & C_y^{(2)} &:= C_y^{(1)}, \end{aligned} \tag{8.8}$$

where

$$w_i := \sqrt{\left(C_x^{(i)} - c_x^{(i)}\right)^2 + \left(C_y^{(i)} - c_y^{(i)}\right)^2}$$

is the diagonal length of ith rectangular area. Notice that the error is now lower (see Figure 27b and Table 13). Therefore, we should try to make another step in the same direction. This time, the expansion is greater, i.e., we set

$$\begin{aligned} c_x^{(3)} &:= c_x^{(2)} - 0.08w_2 \doteq -0.16, \qquad & C_x^{(3)} &:= C_x^{(2)}, \\ c_y^{(3)} &:= c_y^{(2)} - 0.08w_2 \doteq -0.16, \qquad & C_y^{(3)} &:= C_y^{(2)}. \end{aligned} \tag{8.9}$$

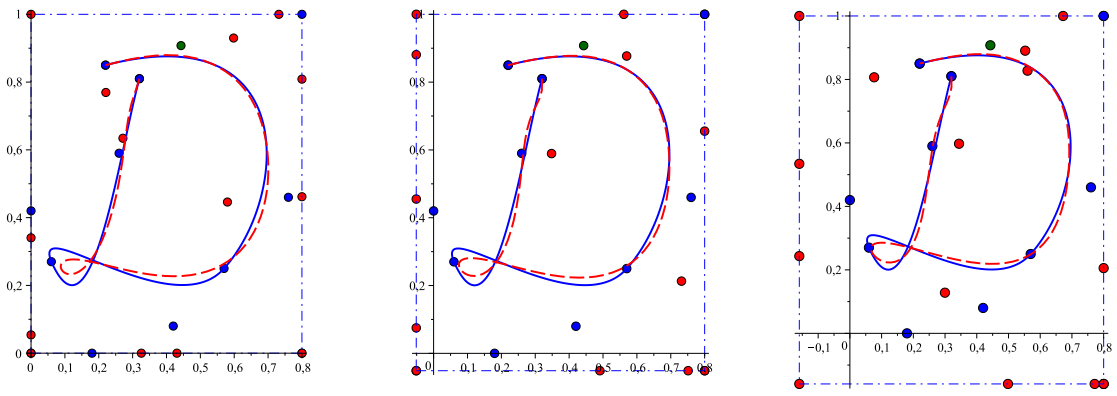
The resulting curve can be seen in Figure 27c. See also Table 13. Observe that, in Example 8.6, the rectangular area (8.6) is even larger. See Figure 25d. Taking into account that `QPSolve` is an iterative method which we apply separately for each coordinate, pairs of numbers of iterations are also given in Table 13.

According to the experiments, if control points of a curve being the solution of Problem 1.10 are located very far away from the plot of the curve (see Figures 23c, 25c and 26c), then it is difficult to find a satisfying solution of Problem 8.2. For that reason, the examples given in this chapter are much more demanding than the ones presented in Chapter 5. Moreover, note that in the case of the box-constrained merging, majority of the resulting control points are located on borders (see Figures 24, 25d and 26d).

Regardless of choice of the rectangular area, one should realize that because of the additional constraints (8.3), approximation error must be inevitably larger than for the traditional approach.

Box constraints	E	E_∞	Iterations
(8.7)	$2.21e{-2}$	$5.56e{-2}$	$(18, 20)$
(8.8)	$1.80e{-2}$	$4.21e{-2}$	$(21, 18)$
(8.9)	$1.42e{-2}$	$3.28e{-2}$	$(19, 26)$
(8.6)	$1.28e{-2}$	$3.01e{-2}$	$(20, 29)$

Table 13: L_2-errors, maximum errors and numbers of iterations for merging of three segments of the composite Bézier curve "D" with box constraints. Parameters: $m = 18$, $k = 0$, $l = 1$.



(a) (b) (c)

Figure 27: Merging of three segments of the composite Bézier curve. The original composite curve (blue solid line with blue control points) and the merged curve (red dashed line with red and green control points) satisfying the following box constraints (blue dotted-dashed frames): (8.7) (see Figure (a)), (8.8) (see Figure (b)) and (8.9) (see Figure (c)). The control points which are constrained by the continuity conditions are green, while the other ones are red and bounded by the blue dotted-dashed frames. Parameters: $m = 18$, $k = 0$, $l = 1$.

Remark 8.9. To solve the box-constrained quadratic programming problem (8.4), one can choose a method provided by software library of selected programming language or implement one of the algorithms given in [39, 49, 80]. For that reason, the running times strongly depend on the implementation of the selected method. However, regardless of the choice, the box constraints make Problem 8.2 more difficult to solve. Therefore, the running times of methods dealing with the new problem must be longer than in the case of the traditional merging. See the comparison given in Table 14.

	Problem 1.10	Problem 8.2	
	Running times [ms]	Running times [ms]	Iterations
Example 8.1	30	227	$(16, 18)$
Example 8.6	48	427	$(20, 29)$
Example 8.7	71	699	$(18, 27)$

Table 14: Running times of the traditional (see Algorithm 6.8) and box-constrained merging of Bézier curves.

Bibliography

[1] Y. J. Ahn, Using Jacobi polynomials for degree reduction of Bézier curves with C^k-constraints, Computer Aided Geometric Design 20 (2003), 423–434.

[2] Y. J. Ahn, B. Lee, Y. Park, J. Yoo, Constrained polynomial degree reduction in the L_2-norm equals best weighted Euclidean approximation of Bézier coefficients, Computer Aided Geometric Design 21 (2004), 181–191.

[3] R. Ait-Haddou, M. Bartoň, Constrained multi-degree reduction with respect to Jacobi norms, Computer Aided Geometric Design 42 (2016), 23–30.

[4] D. Bakhshesh, M. R. Samiee, The weighted dual functions for Wang-Said type generalized Ball bases with and without boundary constraints, International Journal of Computer and Electrical Engineering 4 (2012), 573–577.

[5] R. E. Barnhill, R. F. Riesenfeld (Eds.), Computer Aided Geometric Design, Academic Press, New York, 1974.

[6] B. A. Barsky, T. D. DeRose, Geometric continuity of parametric curves. Technical report UCB/CSD 84/205, University of California, Berkeley, 1984.

[7] M. Bartoň, B. Jüttler, Computing roots of polynomials by quadratic clipping, Computer Aided Geometric Design 24 (2007), 125–141.

[8] S. Basu, R. Pollack, M. F. Roy, Algorithms in Real Algebraic Geometry, second edition, Springer, Berlin, 2006.

[9] P. Bézier, Définition numérique des courbes et surfaces I, Automatisme XI (1966), 625–632 (*in French*).

[10] P. Bézier, Définition numérique des courbes et surfaces II, Automatisme XII (1967), 17–21 (*in French*).

[11] P. Bézier, Procédé de définition numérique des courbes et surfaces non mathématiques, Automatisme XIII (1968), 189–196 (*in French*).

[12] W. Boehm, A. Müller, On de Casteljau's algorithm, Computer Aided Geometric Design 16 (1999), 587–605.

[13] P. Bogacki, S. E. Weinstein, Y. Xu, Degree reduction of Bézier curves by uniform approximation with endpoint interpolation, Computer-Aided Design 27 (1995), 651–661.

[14] J. F. Bonnans, J. C. Gilbert, C. Lemarechal, C. A. Sagastizábal, Numerical Optimization: Theoretical and Practical Aspects, second edition, Springer, Berlin, 1997.

[15] G. Brunnett, T. Schreiber, J. Braun, The geometry of optimal degree reduction of Bézier curves, Computer Aided Geometric Design 13 (1996), 773–788.

[16] G. Chang, T. W. Sederberg, Over and Over Again, first edition, The Mathematical Association of America, Washington, DC, 1997.

[17] X. Chen, W. Ma, J. Paul, Multi-degree reduction of Bézier curves using reparameterization, Computer-Aided Design 43 (2011), 161–169.

[18] G. Chen, G. Wang, Optimal multi-degree reduction of Bézier curves with constraints of endpoints continuity, Computer Aided Geometric Design 19 (2002), 365–377.

[19] J. Chen, G. Wang, Approximate merging of B-spline curves and surfaces, Applied Mathematics-A Journal of Chinese Universities 25 (2010), 429–436.

[20] M. Cheng, G. Wang, Approximate merging of multiple Bézier segments, Progress in Natural Science 18 (2008), 757–762.

[21] Z. Ciesielski, The basis of B-splines in the space of algebraic polynomials, Ukrainian Mathematical Journal 38 (1987), 311–315 (*in Russian*).

[22] M. Daniel, J. C. Daubisse, The numerical problem of using Bézier curves and surfaces in the power basis, Computer Aided Geometric Design 6 (1989), 121–128.

[23] K. R. Davidson, A. P. Donsig, Real Analysis with Real Applications, Prentice Hall, Inc., Upper Saddle River, 2002.

[24] P. de Casteljau, Outillage méthodes calcul. Technical report, André Citroën Automobile SA, Paris, 1959 (*in French*).

[25] P. de Casteljau, Courbes et surfaces à pôles. Technical report, André Citroën Automobile SA, Paris, 1963 (*in French*).

[26] P. de Casteljau, De Casteljau's autobiography: My time at Citroën, Computer Aided Geometric Design 16 (1999), 583–586.

[27] T. D. DeRose, Geometric continuity: a parameterization independent measure of continuity for computer aided geometric design, Ph.D. thesis, University of California, Berkeley, 1985. Available as technical report UCB/CSD 86/255.

[28] E. H. Doha, A. H. Bhrawy, M. A. Saker, On the Derivatives of Bernstein Polynomials: An Application for the Solution of High Even-Order Differential Equations, Boundary Value Problems 2011 (2011), 1–16.

[29] Z. Dostál, Optimal Quadratic Programming Algorithms. With Applications to Variational Inequalities, Springer, New York, 2009.

[30] M. Eck, Degree reduction of Bézier curves, Computer Aided Geometric Design 10 (1993), 237–251.

[31] M. Eck, Least squares degree reduction of Bézier curves, Computer-Aided Design 27 (1995), 845–851.

[32] G. E. Farin, Algorithms for rational Bézier curves, Computer-Aided Design 15 (1983), 73–77.

[33] G. E. Farin, Curves and Surfaces for Computer-Aided Geometric Design. A Practical Guide, fifth edition, Academic Press, Boston, 2002.

[34] G. E. Farin, J. Hoschek, M. Kim, Handbook of Computer Aided Geometric Design, first edition, North Holland, Amsterdam, 2002.

[35] R. T. Farouki, On the stability of transformations between power and Bernstein polynomial forms, Computer Aided Geometric Design 8 (1991), 29–36.

[36] R. T. Farouki, The Bernstein polynomial basis: a centennial retrospective, Computer Aided Geometric Design 29 (2012), 379–419.

[37] R. T. Farouki, V. T. Rajan, On the numerical condition of polynomials in Bernstein form, Computer Aided Geometric Design 4 (1987), 191–216.

[38] R. T. Farouki, V. T. Rajan, Algorithms for polynomials in Bernstein form, Computer Aided Geometric Design 5 (1988), 1–26.

[39] L. Fernandes, A. Fischer, J. Júdice, C. Requejo, J. Soares, A block active set algorithm for large-scale quadratic programming with box constraints, Annals of Operations Research 81 (1998), 75–95.

[40] A. R. Forrest, Interactive interpolation and approximation by Bézier polynomials, The Computer Journal 15 (1972), 71–79.

[41] R. N. Goldman, Dual polynomial bases, Journal of Approximation Theory 79 (1994), 311–346.

[42] R. N. Goldman, Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling, Morgan Kaufmann Publishers, San Francisco, 2003.

[43] P. Gospodarczyk, Degree reduction of Bézier curves with restricted control points area, Computer-Aided Design 62 (2015), 143–151.

[44] P. Gospodarczyk, S. Lewanowicz, P. Woźny, $G^{k,l}$-constrained multi-degree reduction of Bézier curves, Numerical Algorithms 71 (2016), 121–137.

[45] P. Gospodarczyk, P. Woźny, Merging of Bézier curves with box constraints, Journal of Computational and Applied Mathematics 296 (2016), 265–274.

[46] P. Gospodarczyk, P. Woźny, A new property of dual bases and its application, preprint, 2015. Available at `http://arxiv.org/abs/1511.08264`.

[47] M. P. Groover, E. W. Zimmers, CAD/CAM: Computer-Aided Design and Manufacturing, Prentice Hall, Inc., Englewood Cliffs, 1983.

[48] Z. Guohui, L. Xiuping, S. Zhixun, A dual functional to the univariate B-spline, Journal of Computational and Applied Mathematics 195 (2006), 292–299.

[49] C. Han, P. M. Pardalos, Y. Ye, Computational aspects of an interior point algorithm for quadratic programming problems with box constraints, in: T. F. Coleman, Y. Li (Eds.), Proceedings of the Workshop on Large-Scale Numerical Optimization, SIAM, Philadelphia, 1990, 92–112.

[50] J. Hoschek, Approximate conversion of spline curves, Computer Aided Geometric Design 4 (1987), 59–66.

[51] S. Hu, R. Tong, T. Ju, J. Sun, Approximate merging of a pair of Bézier curves, Computer-Aided Design 33 (2001), 125–136.

[52] B. Jüttler, The dual basis functions for the Bernstein polynomials, Advances in Computational Mathematics 8 (1998), 345–352.

[53] S. N. Kersey, Dual basis functions in subspaces of inner product spaces, Applied Mathematics and Computation 219 (2013), 10012–10024.

[54] P. Kiciak, Podstawy modelowania krzywych i powierzchni. Zastosowania w grafice komputerowej, wydanie drugie, WNT, Warszawa, 2005 (*in Polish*).

[55] H. J. Kim, A. J. Ahn, Good Degree Reduction of Bézier Curves Using Jacobi Polynomials, Computers and Mathematics with Applications 40 (2000), 1205–1215.

[56] M. K. Kozlov, S. P. Tarasov, L. G. Khachian, Polynomial solvability of convex quadratic programming, Soviet Mathematics Doklady 20 (1979), 1108–1111.

[57] M. Lachance, Approximation by constrained parametric polynomials, Rocky Mountain Journal of Mathematics 21 (1991), 473–488.

[58] K. Lalit Narayan, K. Mallikarjuna Rao, M. M. M. Sarcar, Computer Aided Design and Manufacturing, Prentice-Hall of India Private Limited, New Delhi, 2008.

[59] C. W. Lawson, R. J. Hanson, Solving Least Squares Problems, John Wiley and Sons, Inc., New York, 1974.

[60] B. Lee, Y. Park, J. Yoo, Application of Legendre-Bernstein basis transformations to degree elevation and degree reduction, Computer Aided Geometric Design 19 (2002), 709–718.

[61] S. Lewanowicz, P. Keller, P. Woźny, Constrained approximation of rational triangular Bézier surfaces by polynomial triangular Bézier surfaces, preprint, 2015. Available at `http://arxiv.org/abs/1504.03557`.

[62] S. Lewanowicz, P. Woźny, Dual generalized Bernstein basis, Journal of Approximation Theory 138 (2006), 129–150.

[63] S. Lewanowicz, P. Woźny, Bézier representation of the constrained dual Bernstein polynomials, Applied Mathematics and Computation 218 (2011), 4580–4586.

[64] S. Lewanowicz, P. Woźny, Multi-degree reduction of tensor product Bézier surfaces with general constraints, Applied Mathematics and Computation 217 (2011), 4596–4611.

[65] S. Lewanowicz, P. Woźny, P. Keller, Polynomial approximation of rational Bézier curves with constraints, Numerical Algorithms 59 (2012), 607–622.

[66] S. Lewanowicz, P. Woźny, P. Keller, Weighted polynomial approximation of rational Bézier curves, technical report, 2015. Available at `http://arxiv.org/abs/1502.07877`.

[67] R. A. Liming, Practical analytical geometry with applications to aircraft, Macmillan, New York, 1944.

[68] L. Lu, A note on iterative process for G^2-multi degree reduction of Bézier curves, Applied Mathematics and Computation 218 (2012), 6987–6990.

[69] L. Lu, Effective C^1G^2-merging of Two Bézier Curves by Matrix Computation, International Journal of Advancements in Computing Technology 5 (2013), 1117–1123.

[70] L. Lu, Explicit G^2-constrained degree reduction of Bézier curves by quadratic optimization, Journal of Computational and Applied Mathematics 253 (2013), 80–88.

[71] L. Lu, An explicit method for G^3 merging of two Bézier curves, Journal of Computational and Applied Mathematics 260 (2014), 421–433.

[72] L. Lu, Explicit algorithms for multiwise merging of Bézier curves, Journal of Computational and Applied Mathematics 78 (2015), 138–148.

[73] L. Lu, Gram matrix of Bernstein basis: Properties and applications, Journal of Computational and Applied Mathematics 280 (2015), 37–41.

[74] L. Lu, Some improvements on optimal multi-degree reduction of Bézier curves with geometric constraints, Computer-Aided Design 59 (2015), 39–42.

[75] L. Lu, C. Jiang, An iterative algorithm for G^2 multiwise merging of Bézier curves, Journal of Computational and Applied Mathematics 296 (2016), 352–361.

[76] L. Lu, G. Wang, Optimal multi-degree reduction of Bézier curves with G^2-continuity, Computer Aided Geometric Design 23 (2006), 673–683.

[77] L. Lu, G. Wang, A quadratic programming method for optimal degree reduction of Bézier curves with G^1-continuity, Journal of Zhejiang University SCIENCE A 8 (2007), 1657–1662.

[78] L. Lu, G. Wang, Application of Chebyshev II-Bernstein basis transformations to degree reduction of Bézier curves, Journal of Computational and Applied Mathematics 221 (2008), 52–65.

[79] D. Lutterkort, J. Peters, U. Reif, Polynomial degree reduction in the L_2-norm equals best Euclidean approximation of Bézier coefficients, Computer Aided Geometric Design 16 (1999), 607–612.

[80] J. J. Moré, G. Toraldo, Algorithms for bound constrained quadratic programming problems, Numerische Mathematik 55 (1989), 377–400.

[81] M. Müller-Prove, Vision and Reality of Hypertext and Graphical User Interfaces, master's thesis, Universität Hamburg, Hamburg, 2002. Available at `http://edoc.sub.uni-hamburg.de/informatik/volltexte/2009/52/pdf/B_237.pdf` (accessed July 20, 2015).

[82] M. B. Nathanson, Additive Number Theory: The Classical Bases, Springer, New York, 1996.

[83] Y. Park, U. J. Choi, The Error Analysis for Degree Reduction of Bézier Curves, Computers and Mathematics with Applications 27 (1994), 1–6.

[84] L. Piegl, W. Tiller, Algorithm for degree reduction of B-spline curves, Computer-Aided Design 27 (1995), 101–110.

[85] L. Piegl, W. Tiller, The NURBS Book, second edition, Springer, Berlin, 1997.

[86] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-Spline Techniques, first edition, Springer, Berlin, 2002.

[87] A. Rababah, M. Al-Natour, The weighted dual functionals for the univariate Bernstein basis, Applied Mathematics and Computation 186 (2007), 1581–1590.

[88] A. Rababah, M. Al-Natour, Weighted dual functions for Bernstein basis satisfying boundary constraints, Applied Mathematics and Computation 199 (2008), 456–463.

[89] A. Rababah, B. Lee, J. Yoo, A simple matrix form for degree reduction of Bézier curves using Chebyshev-Bernstein basis transformations, Applied Mathematics and Computation 181 (2006), 310–318.

[90] A. Rababah, B. Lee, J. Yoo, Multiple Degree Reduction and Elevation of Bézier Curves Using Jacobi-Bernstein Basis Transformations, Numerical Functional Analysis and Optimization 28 (2007), 1179–1196.

[91] A. Rababah, S. Mann, Iterative process for G^2-multi degree reduction of Bézier curves, Applied Mathematics and Computation 217 (2011), 8126–8133.

[92] A. Rababah, S. Mann, Linear methods for G^1, G^2, and G^3–Multi-degree reduction of Bézier curves, Computer-Aided Design 45 (2013), 405–414.

[93] P. B. Stark, R. L. Parker, Bounded-Variable Least-Squares: an Algorithm and Applications, Computational Statistics 10 (1995), 129–141.

[94] H. Sunwoo, Matrix representation for multi-degree reduction of Bézier curves, Computer Aided Geometric Design 22 (2005), 261–273.

[95] H. Sunwoo, N. Lee, A unified matrix representation for degree reduction of Bézier curves, Computer Aided Geometric Design 21 (2004), 151–164.

[96] C. Tai, S. Hu, Q. Huang, Approximate merging of B-spline curves via knot adjustment and constrained optimization, Computer-Aided Design 35 (2003), 893–899.

[97] M. A. Watkins, A. J. Worsey, Degree reduction of Bézier curves, Computer-Aided Design 20 (1988), 398–405.

[98] D. E. Weisberg, The engineering design revolution, 2008. Available at `http://www.cadhistory.net/toc.htm` (accessed July 20, 2015).

[99] P. Wolfe, The simplex algorithm for quadratic programming, Econometrica 27 (1959), 382–398.

[100] P. Woźny, Construction of dual bases, Journal of Computational and Applied Mathematics 245 (2013), 75–85.

[101] P. Woźny, Construction of dual B-spline functions, Journal of Computational and Applied Mathematics 260 (2014), 301–311.

[102] P. Woźny, P. Gospodarczyk, S. Lewanowicz, Efficient merging of multiple segments of Bézier curves, Applied Mathematics and Computation 268 (2015), 354–363.

[103] P. Woźny, S. Lewanowicz, Multi-degree reduction of Bézier curves with constraints, using dual Bernstein basis polynomials, Computer Aided Geometric Design 26 (2009), 566–579.

[104] P. Woźny, S. Lewanowicz, Constrained multi-degree reduction of triangular Bézier surfaces using dual Bernstein polynomials, Journal of Computational and Applied Mathematics 235 (2010), 785–804.

[105] Y. Ye, E. Tse, An extension of Karmarkar's projective algorithm for convex quadratic programming, Mathematical Programming 44 (1989), 157–179.

[106] L. Zhang, J. Tan, Z. Dong, The dual bases for the Bézier-Said-Wang type generalized Ball polynomial bases and their applications, Applied Mathematics and Computation 217 (2010), 3088–3101.

[107] L. Zhang, J. Tan, H. Wu, Z. Liu, The weighted dual functions for Wang-Bézier type generalized Ball bases and their applications, Applied Mathematics and Computation 215 (2009), 22–36.

[108] L. Zhang, H. Wu, J. Tan, Dual bases for Wang-Bézier basis and their applications, Applied Mathematics and Computation 214 (2009), 218–227.

[109] L. Zhang, H. Wu, J. Tan, Dual basis functions for the NS-power basis and their applications, Applied Mathematics and Computation 207 (2009), 434–441.

[110] R. Zhang, G. Wang, Constrained Bézier curves' best multi-degree reduction in the L_2-norm, Progress in Natural Science 15 (2005), 843–850.

[111] R. Zhang, G. Wang, W. Ma, Best Multi-degree Reduction of Bernstein Polynomial in L_2-norm Based on an Explicit Termination Criterion, Computer-Aided Design & Applications 4 (2007), 181–190.

[112] L. Zhou, G. Wang, Optimal constrained multi-degree reduction of Bézier curves with explicit expressions based on divide and conquer, Journal of Zhejiang University SCIENCE A 10 (2009), 577–582.

[113] L. Zhou, G. Wang, Matrix representation for optimal multi-degree reduction of Bézier curves with G^1 constraints, Journal of Computer Aided Design & Computer Graphics 22 (2010), 735–740.

[114] L. Zhou, Y. Wei, Y. Yao, Optimal multi-degree reduction of Bézier curves with geometric constraints, Computer-Aided Design 49 (2014), 18–27.

[115] P. Zhu, G. Wang, Optimal approximate merging of a pair of Bézier curves with G^2-continuity, Journal of Zhejiang University SCIENCE A 10 (2009), 554–561.

Index