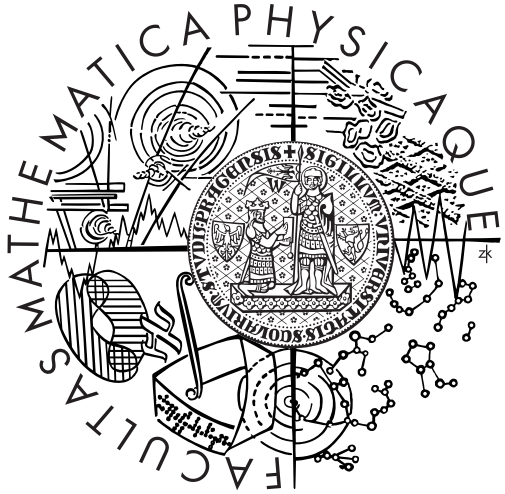
Charles University in Prague
Faculty of Mathematics and Physics

Doctoral Thesis



Michael Bartoň

Quadratic Clipping and its Generalization for Polynomial Systems

Supervisor: *Prof. RNDr. Adolf Karger, DrSc.*

Field of Study: *General Problems of Mathematics and Computer Science*

Acknowledgements

I would like to thank Prof. Adolf Karger, my supervisor, for his suggestions and constant support during this research. I am also thankful to Prof. Bert Jüttler for his guidance through my short term attachment at Johann Radon Institute for Computational and Applied Mathematics in Linz, Austria, which was crucial to the successful completion of this thesis.

I would also like to thank the Austrian Science Fund (FWF) for partially supporting this research through the project SFB F013 "Numerical and Symbolic Scientific Computing", subproject 15.

I hereby certify that I have written this thesis myself without using any sources other than cited.

In Prague, January 18, 2007 Michael Bartoň

Abstract

We present an algorithm which is able to compute all roots of a given univariate polynomial within a given interval. In each step, we use degree reduction to generate a strip bounded by two quadratic polynomials which encloses the graph of the polynomial within the interval of interest. The new interval(s) containing the root(s) is (are) obtained by intersecting this strip with the abscissa axis. In the case of single roots, the sequence of the lengths of the intervals converging towards the root has the convergence rate 3. For double roots, the convergence rate is still superlinear ($\frac{3}{2}$). We show that the new technique compares favorably with the classical technique of Bézier clipping.

The generalization of algorithm for polynomial systems is presented. Demonstrating on bivariate case, polynomials are both represented by surfaces in $\mathbb{R}^3$ and the solution is therefore the intersection of graphs of both polynomials and the plane xy. Likewise the univariate case, we construct the linear bounding spatial strips of graphs of both polynomials, which define a parallelogram in the plane xy. This parallelogram is intersected with the original domain in order to define the new one.

Keywords

root finding, polynomial, Bézier clipping, polynomial system, L^2 norm.

Contents

Acknowledgements ii

Abstract iii

1 Introduction 1

2 Preliminaries 6

2.1 Bézier curves 6

2.1.1 The de Casteljau algorithm 11

2.1.2 Degree elevation 14

2.1.3 Nonparametric curves 18

2.2 Bézier patches 20

2.2.1 Tensor product 24

2.2.2 Degree elevation 26

2.2.3 Nonparametric patches 27

2.3 Linear space of polynomials 29

2.4 Degree reduction and dual basis 32

2.5 Bézier clipping and its convergence rate 35

3 Computing roots via quadratic clipping **40**

3.1 The root–finding problem 40

3.2 Algorithm 41

3.3 Convergence rate 46

4 Comparison **53**

4.1 Convergence rates, number of operations and time per iteration step 53

4.2 Number of iterations and computing times vs. accuracy 55

5 Bivariate linear clipping **66**

5.1 The root–finding problem 66

5.2 The generalization of L_2 norm and dual basis 67

5.2.1 L_2 norm 67

5.2.2 Dual basis 68

5.3 Algorithm 70

5.4 Examples 75

6 Conclusion **84**

List of Figures **85**

List of Tables **91**

References **93**

1 Introduction

Efficient and robust algorithms which compute the solutions (of systems of) polynomial equations are frequently needed for modeling, processing and visualizing free-form geometry described by piecewise rational parametric representations. For instance, the problem of intersecting a straight line with a rational parametric surface leads to a polynomial system consisting of two equations for two unknowns. If the surface is given in implicit form, then only a single equation has to be solved. Such intersections have to be computed for visualizing free–form surfaces using ray-tracing Nishita, Sederberg and Kakimoto (1990); Efremov, Havran and Seidel (2005). Similarly, the problem of computing the closest point(s) on a curve or surface to a given point leads to polynomial equations, see e.g. Wang, Kearney and Atkinson (2003).

Solutions of various geometric problems in computer geometry, such as surface–surface intersections, bisectors / medial axes, convex hull computations, etc., lead to piecewise algebraic curves Lee (1999); Patrikalakis and Maekawa (2002b); Kim, Elber and Seong (2005). In this situation, efficient methods for analyzing and representing these curves are needed Gonzalez-Vega and Necula (2002); Gatellier et al. (2003). Root finding algorithms for (systems of) polynomial equations are again an important ingredient of these techniques; they are used to determine "critical points" (which are needed to determine the topology of the curve) and suitable initial points for tracing the curve.

More precisely, in these and similar applications, all solutions of a (system of) polynomial equation(s) within a certain domain Ω, which is typically a box in $\mathbb{R}^n$, are sought for. We are interested in *numerical* techniques which guarantee that *all* solutions are found. Two major approaches exist:

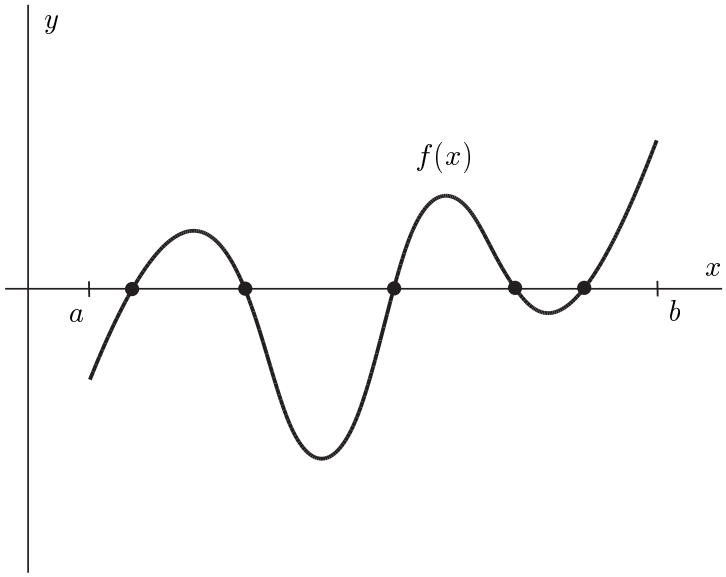


Figure 1: Root-finding problem: all roots of polynomial $f(x)$ on interval $[a, b]$ are required.

Homotopy methods (see, e.g. Li, 2003; Sommese and Wampler, 2005) start with the solutions of a simpler system with the same structure of the set of solutions. This system is then continuously transformed into the original system, and the solutions are found by tracing the solutions of the auxiliary system. These techniques are particularly well suited for $\Omega = \mathbb{C}^n$.

Bézier clipping and related techniques are based on the convex-hull property of Bernstein-Bézier- (BB-) representations. The main idea is described in Section 2.5. Combined with subdivision, these techniques lead to fast (achieving quadratic convergence for single roots) solvers for univariate polynomials Nishita, Sederberg and Kakimoto (1990); Nishita and Sederberg (1990). Multivariate versions, such as the IPP algorithm, exist and have found their way

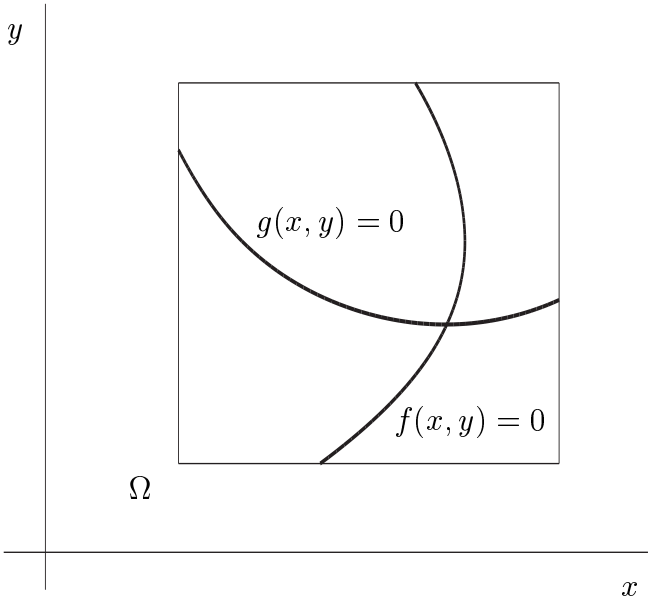


Figure 2: Bivariate polynomial system: polynomial system (1) over domain Ω represented as curves intersection.

into industrial software, such as commercial CAD systems Sheerbrooke and Patrikalakis (1993); Elber and Kim (2001); Ko, Sakkalis and Patrikalakis (2005); Mourrain and Pavone (2005).

We will formulate a novel technique for computing the roots of univariate polynomials and will present a way of a generalization for multivariate case. Both methods are based on *degree reduction*. This term denotes the process of approximating a polynomial of a certain degree by a lower degree one, with respect to a suitable norm, and possibly subject to boundary conditions. It has been studied thoroughly in the rich literature on this subject Eck (1992); Lutterkort, Peters and Reif (1999); Ahn, Lee, Park and Yoo (2004); Sunwoo (2005). In earlier years, the issue of degree reduction was motivated by degree

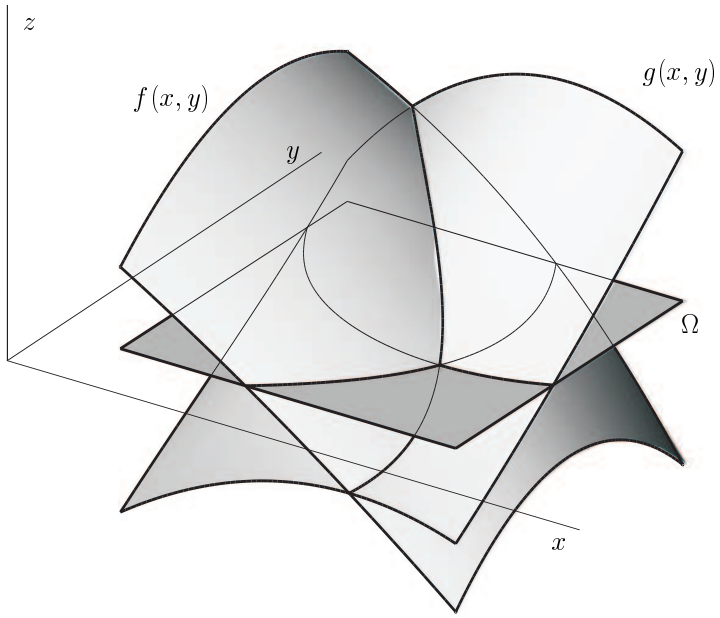


Figure 3: Bivariate polynomial system: solution of the system (1) represented as the intersection of three surfaces: $z = f(x, y)$, $z = g(x, y)$, $z = 0$.

limitations of CAD systems.

The basic idea of the new robust univariate root solver quadratic clippinq (`quadclip`) is briefly descibed as follows. The input of the algorithm consists of a polynomial f, interval $[a, b]$ and accuracy ε, the output are all roots of f on $[a, b]$ within desired precision ε. In the first step, the best quadratic approximant of f with respect to L^2 norm is detected. Then, upper and lower quadratic bounds are constructed and its roots symbolically computed. At most four real roots are obtained. Discussing the mutual position of roots and the original interval $[a, b]$, a new subinterval is located. If no roots exist, the interval is erased. Algorithm works till the lenght of all root-containing

intervals is less then prescribed accuracy ε.

The possibility of generalization of `quadclip` for multivariate polynomial systems is shown. Demonstrated on bivariate system

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0, \end{aligned} \tag{1}$$

the algorithm `linbivar` works with three surfaces in $\mathbb{R}^3$ instead of two curves in $\mathbb{R}^2$ (compare: Figure 2 and 3). Requiring all roots of system (1) lying in domain Ω, the best planar approximants with respect to generalized L^2 norm of both surfaces are found. Constructing planar boundaries for both surfaces, its intersection with plane (x, y) gives the new domain.

The remainder of this thesis is organized as follows. Section 2 summarizes the basic informations on Bézier curves and patches, degree reduction, dual basis, L^2 norm and about the classical technique of Bézier clipping.

Section 3 desribes the new algorithm: quadratic clippinq (`quadclip`) and shows cubical convergence rate in single roots and superlinear convergence rate in double root cases. Further, we provide a detailed comparison of `quadclip` with Bézier clipping with respect to criteria such as computational effort, rate of convergence and computing times.

Section 4 shows the generalized algorithm (`linbivar`) for bivariate polynomial systems (1). We present roughcasted comparison with results of Mourrain and Pavone (2005). In addition, the way of generalization for system of n variables is briefly suggested.

2 Preliminaries

In this section, we summarize some basic concepts and results concerning Bézier curves and patches, dual basis, degree reduction, L^2 norm and Bézier clipping.

2.1 Bézier curves

A Bézier curve of degree $n \in \mathbb{N}$ is defined by formula

$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{b}_i, \quad t \in [0,1], \tag{2}$$

where coefficients $\mathbf{b}_i \in \mathbb{E}_m$ are called *control points* and $B_i^n(t)$ are the *Bernstein polynomials* given by

$$B_i^n(t) = \binom{n}{i} t^i (t^{n-i}), \tag{3}$$

and the binomial coefficients are determined by

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if} \quad 0 \le i \le n \\ 0 & \text{otherwise.} \end{cases}$$

Example 1 Bézier curves of degree 4 of different shape are shown in Fig. 5, corresponding Bernstein polynomials

$$B_0^4(t) = (1-t)^4, \tag{5}$$
$$B_1^4(t) = 4t(1-t)^3, \tag{6}$$
$$B_2^4(t) = 6t^2(1-t)^2, \tag{7}$$
$$B_3^4(t) = 4t^3(1-t), \tag{8}$$
$$B_4^4(t) = t^4, \tag{9}$$

are displayed in Fig. 4.

Observing the importance of the order of control points, two neighbouring points are connected by a line. The set of these lines and control points is known as *control polygon*.

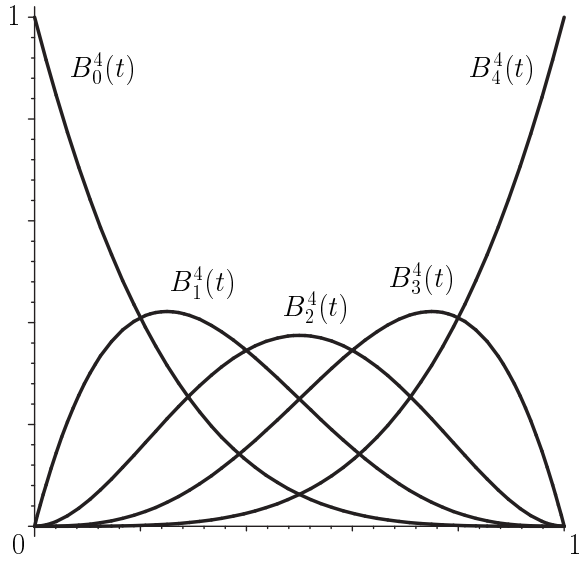


Figure 4: Bernstein polynomials for $n = 4$.

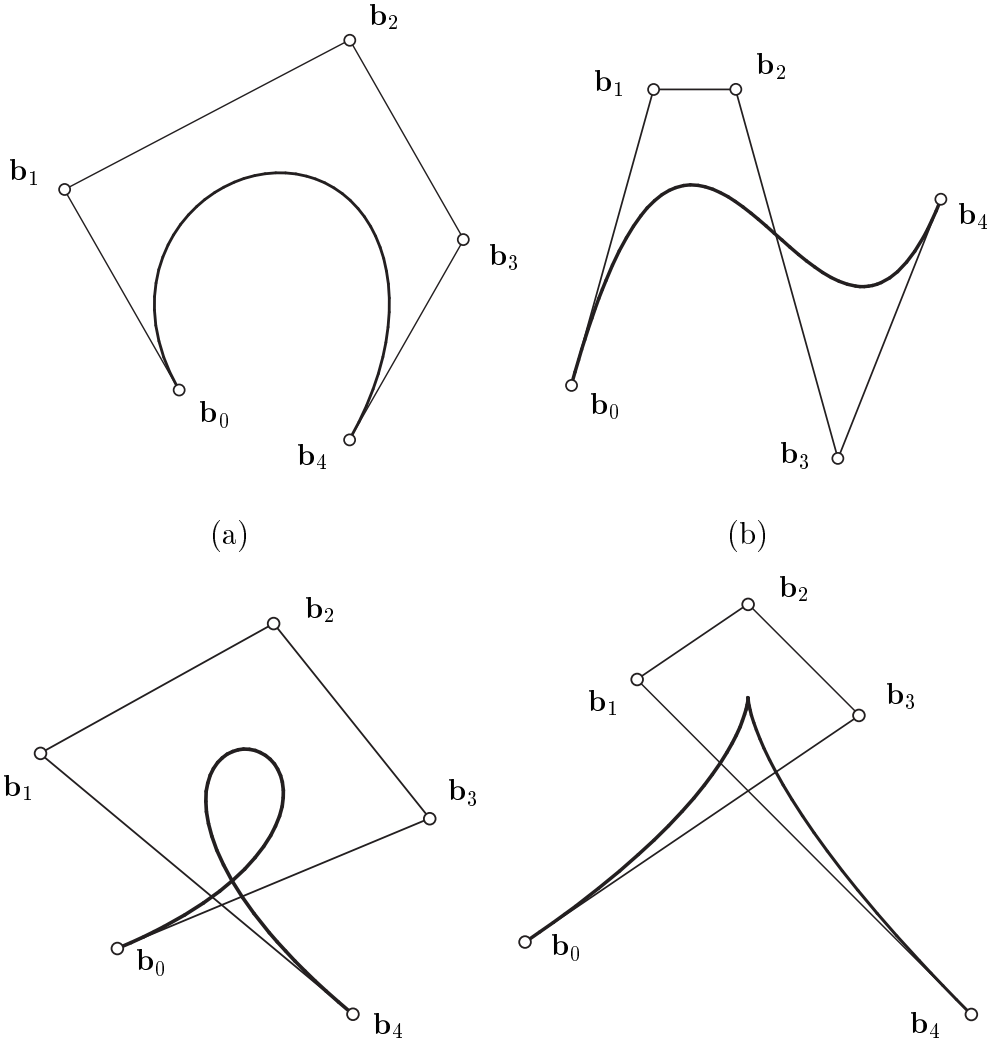


Figure 5: Bézier curves of degree 4: curve with b) inflection, c) loop, d) cusp.

Properties of Bernstein polynomials: Looking at the definition of Bernstein polynomials in more detail, the definition (3) yields directly the list of properties below:

1. nonnegativity: $B_i^n(t) \geq 0$ for all i, n and $t \in [0,1]$;
2. partition of unity: $\sum_{i=0}^{n} B_i^n(t) = 1$ for all $t \in [0,1]$;
3. linear precision: $\sum_{i=0}^{n} \frac{i}{n} B_i^n(t) = t$ for all $t \in [0,1]$;
4. symmetry: $B_i^n(t) = B_{n-i}^n(1-t)$ for any n and $t \in [0,1]$;
5. recursion: $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$;
6. for any n, $B_i^n(t)$ attains exactly one maximum on $[0,1]$ for $t = \frac{i}{n}$;
7. derivative: $\frac{d}{dt}B_i^n(t) = n(B_{i-1}^{n-1}(t) + B_i^{n-1}(t))$, where $B_{-1}^{n-1}(t) \equiv B_n^{n-1}(t) \equiv 0$;
8. $(1-t)B_i^n(t) = \frac{n+1-i}{n+1}B_i^{n+1}(t)$;
9. $tB_i^n(t) = \frac{i+1}{n+1}B_{i+1}^{n+1}(t)$;

Properties of Bézier curves: As a result of Bernstein polynomials properties above, one can easily formulate following attributes of Bézier curves.

1. affine invariance: barycentric combinations are invariant under affine mapping. In other words, affine image of control polygon and the control polygon of the affine image of the curve are the same;
2. convex hull property: for any Bézier curve and any parameter $t_0 \in [0,1]$, the point $\mathbf{b}(t_0)$ lies inside the convex hull of the control polygon;
3. endpoints interpolation: $\mathbf{b}(0) = \mathbf{b}_0$ and $\mathbf{b}(1) = \mathbf{b}_n$;

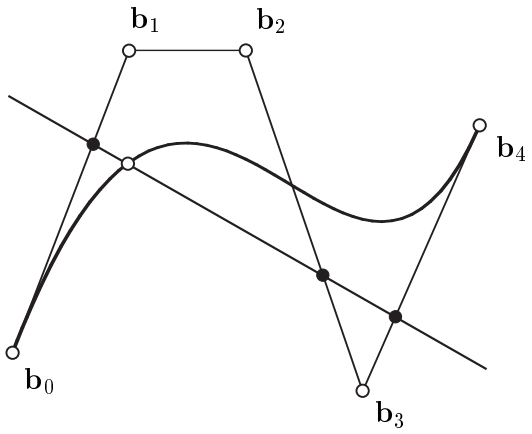


Figure 6: Variation diminishing property: arbitrary line has no more intersections with Bézier curve then with its control polygon.

4. end-tangent vectors: $\mathbf{b}'(0) = n(\mathbf{b}_1 - \mathbf{b}_0)$ and $\mathbf{b}'(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1})$;

5. variation diminishing property: number of intersections of a line with Bézier curve is bounded by the number of intersections of the line with control polygon.

Example 2 Let $\mathbf{b}_0$, $\mathbf{b}_1$, $\mathbf{b}_2$ be three control points of Bézier curve of degree $n = 2$, $\mathbf{b}(t) = \sum_{i=0}^{2} B_i^2 \mathbf{b}_i$, $t \in [0, 1]$ and let us construct

$$\begin{aligned} \mathbf{b}_0^1(t) &= (1-t)\mathbf{b}_0 + t\mathbf{b}_1, \\ \mathbf{b}_1^1(t) &= (1-t)\mathbf{b}_1 + t\mathbf{b}_2, \end{aligned}$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1.$$

Inserting the first two equations into the third one, we obtain

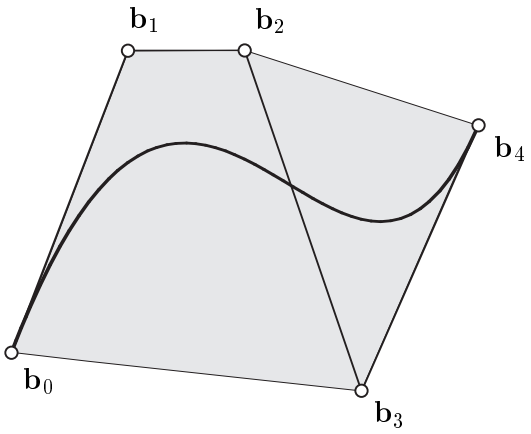


Figure 7: Convex hull property: curve is included inside the convex hull of its control polygon.

$$\mathbf{b}(t) = \sum_{i=0}^{2} B_i^2 \mathbf{b}_i = \mathbf{b}_0^2(t).$$

We observe that given parabola is decomposed into two linear interpolants. Therefore, for fixed value $t_0 \in [0, 1]$, it is easy to construct $\mathbf{b}(t_0)$ by three linear interpolations (see Figure 8).

2.1.1 The de Casteljau algorithm

The basic idea of linear decomposition, shown in example 2, is easily generalized to arbitrary degree. Let us assume Bézier curve of degree n, $\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n \mathbf{b}_i$. Then, with respect to recursive property of Bernstein polynomials, we obtain

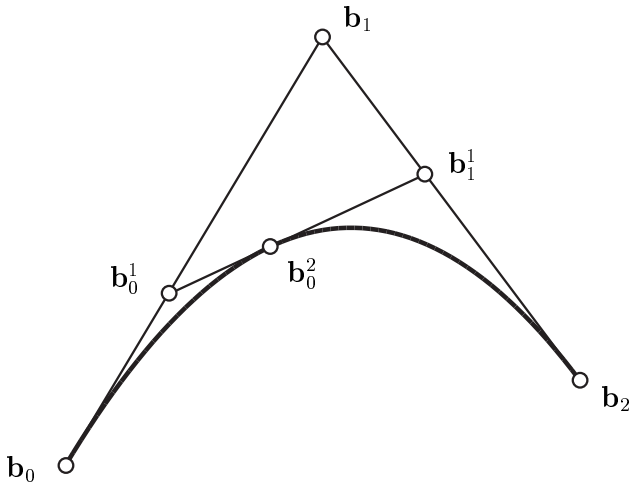


Figure 8: Construction of parabola by linear interpolation.

$$\mathbf{b}(t) = (1-t)\sum_{i=0}^{n-1} B_i^{n-1}\mathbf{b}_i + t\sum_{i=0}^{n-1} B_i^{n-1}\mathbf{b}_{i+1}. \tag{10}$$

In other words, $\mathbf{b}(t)$ is expressed by linear combination of two curves of degree $n-1$. This principle is a basic ground for the de Casteljau algorithm (11), which recursively repeats the decomposition. For a fixed value $t = t_0$, we obtain a recursive formula for constructing the point $\mathbf{b}(t_0) = \mathbf{b}_0^n$ on given Bézier curve:

$$\mathbf{b}_i^k(t_0) = (1-t_0)\mathbf{b}_i^{k-1}(t_0) + t_0\mathbf{b}_{i+1}^{k-1}(t_0), \tag{11}$$

where $\mathbf{b}_i = \mathbf{b}_0^i$, $k = 1, 2, \ldots, n$ and $i = 0, 1, \ldots, n-k$. Sought point $\mathbf{b}(t_0)$ is found by sequence of linear interpolations (see table 1 and figure 9).

Subdivision: A Bézier curve $\mathbf{b}^n$ is usually defined on the interval $[0, 1]$, but it can be easily defined on arbitrary subinterval $[0, c]$. Handling Bézier curve

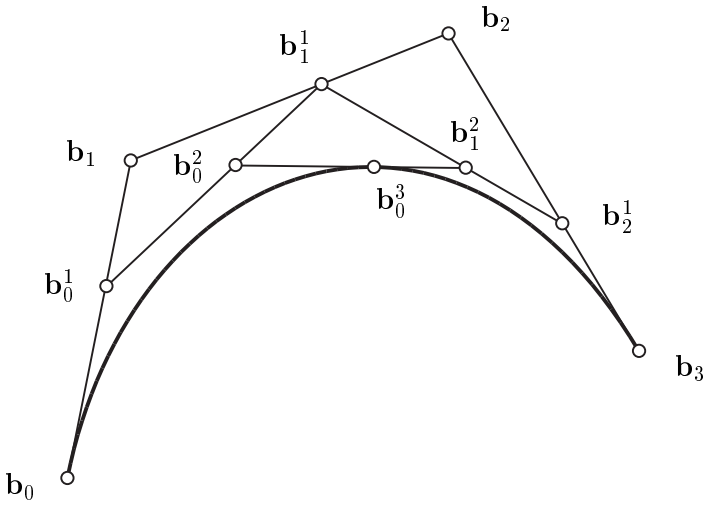


Figure 9: de Casteljau algorithm: Construction of a point of Bézier curve for parameter value $t_0 = \frac{3}{5}$.

by control points, the construction of the new control polygon is required. Finding this control polygon is known as *subdivision* of Bézier curve.

Using de Casteljau algorithm, the control points $\mathbf{c}_i$, that define curve $\mathbf{b}^n$ on subinterval $[0, c]$, are given by formula

$$\mathbf{c}_i = \mathbf{b}_0^i(c), \tag{12}$$

for all $i = 0, \ldots, n$, (see Fig. 10). Analogously, following the symmetry property of Bernstein polynomials, the control points corresponding to the subinterval $[c, 1]$ are given by the $\mathbf{b}_i^{n-i}(c)$, $i = 0, \ldots, n$.

$$\begin{array}{ccccccccccc}
\mathbf{b}_0 & & \mathbf{b}_1 & & \mathbf{b}_2 & \dots & \dots & \dots & \mathbf{b}_{n-2} & \mathbf{b}_{n-1} & \mathbf{b}_n \\
& \mathbf{b}_0^1 & & \mathbf{b}_1^1 & & \mathbf{b}_2^1 & \dots & \dots & \mathbf{b}_{n-2}^1 & \mathbf{b}_{n-1}^1 & \mathbf{b}_n^1 \\
& & \mathbf{b}_0^2 & & \mathbf{b}_1^2 & \dots & \dots & \dots & \mathbf{b}_{n-1}^2 & \mathbf{b}_n^2 & \\
& & & & & & \vdots & & & & \\
& & & & \mathbf{b}_0^{n-1} & & & & \mathbf{b}_1^{n-1} & & \\
& & & & & & \mathbf{b}_0^n & & & &
\end{array}$$

Table 1: Scheme of point construction via de Casteljau algorithm.

2.1.2 Degree elevation

Handling and shaping a Bézier curve using its control points is really a skillful approach. With respect to barycentric coordinates, each control point has its *weight* of influence on changing the shape. In order to modify the curve slightly, more control points are desired. *Degree elevation* (or degree raising) is a standard technique, which allows us to describe a n-th degree Bézier curve as a Bézier curve of higher degree.

Let us assume Bézier curve of degree n is given by its control points $\mathbf{b}_0, \dots, \mathbf{b}_n$. Our task is to find a control polygon $\mathbf{b}_0^{(1)}, \dots, \mathbf{b}_{n+1}^{(1)}$ that defines the same curve.

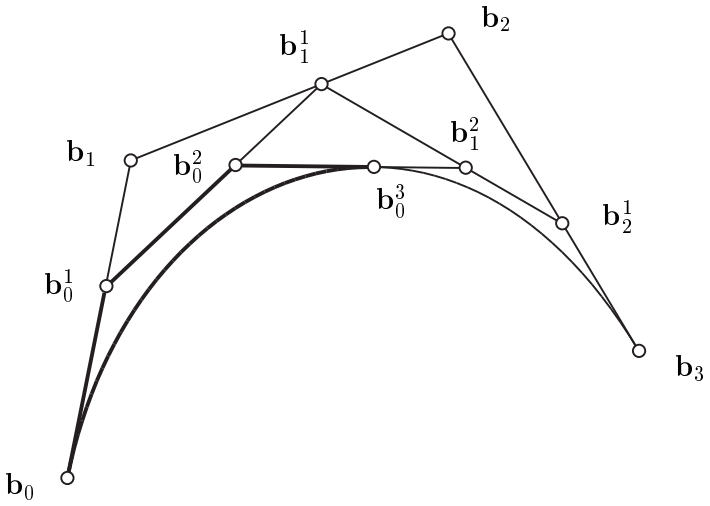


Figure 10: Subdivision: Construction of a control polygon of Bézier curve defined on subinterval $[0, \frac{3}{5}]$.

According to the notation from (2) and using properties of Bernstein polynomials we rewrite our curve as

$$\mathbf{b}(t) = \sum_{i=0}^{n} \frac{n+1-i}{n+1} B_i^{n+1}(t)\mathbf{b}_i + \sum_{i=0}^{n} \frac{i+1}{n+1} B_{i+1}^{n+1}(t)\mathbf{b}_i, \quad t \in [0,1]. \tag{13}$$

Looking at this equation in more detail, one can easily see that the upper bound of the first sum may be increased to $n+1$ since the last expression is equal to zero. Shifting the index of the second sum by one and adding a zero-term for $i = 0$ we get

$$\mathbf{b}(t) = \sum_{i=0}^{n+1} \frac{n+1-i}{n+1} B_i^{n+1}(t)\mathbf{b}_i + \sum_{i=0}^{n+1} \frac{i+1}{n+1} B_i^{n+1}(t)\mathbf{b}_{i-1}, \quad t \in [0,1]. \tag{14}$$

Summing up appropriate coefficients, we obtain a formula for control points of $(n+1)$-st degree Bézier curve

$$\mathbf{b}_i^{(1)} = \frac{i}{n+1}\mathbf{b}_{i-1} + (1 - \frac{i}{n+1})\mathbf{b}_i, \quad i = 0, \ldots, n+1. \tag{15}$$

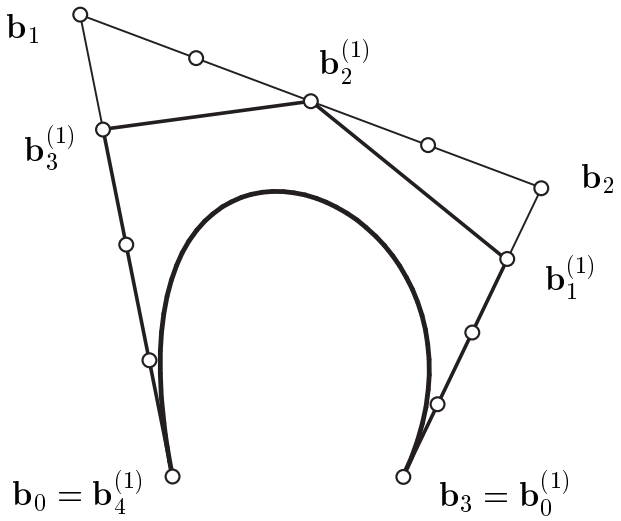


Figure 11: Degree elevation: Bézier cubic represented by five control points after one elevation step.

Vertices $\mathbf{b}_i^{(1)}$ are constructed by piecewise linear interpolation and convex hull of the new control polygon lies inside the original one (see Figure 11).

One can easily see, that this process may be repeated to obtain control polygon with higher number of control points than is the degree of given curve. This act allows us to modify original curve with higher precision which is desirable in many applications.

Let us denote control polygon of n-th degree Bézier curve by $\mathbf{b}_0, \ldots, \mathbf{b}_n$. After r degree elevation steps, this curve is described by $n + r$ control points $\mathbf{b}_0^{(1)}, \ldots, \mathbf{b}_{n+r}^{(1)}$ (see Figure 12). New control points are defined by formula

$$\mathbf{b}_i^{(r)} = \sum_{j=0}^{n} \mathbf{b}_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}}, \tag{16}$$

which is easily proved by induction.

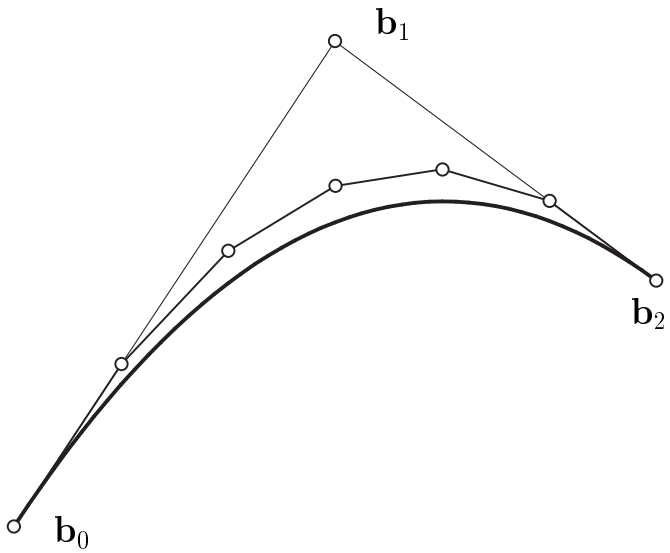


Figure 12: Repeated degree elevation: Parabolic arc after four degree elevation steps.

Degree reduction: The inverse process is at least as important as the previous one. Having a Bézier curve of degree n, the goal is to find Bézier curve of lower degree defining the same curve. Generally, the exact degree reduction is not possible. For example, cubic with a cusp cannot be expressed like quadratic. Thus, degree reduction is an aproximative technique and the value of the error between the original curve and the reduced one depends on the metric we apply. Many techniques are known (for more information see Introduction and References). Reduction with respect to L^2 metric is described in section (2.4).

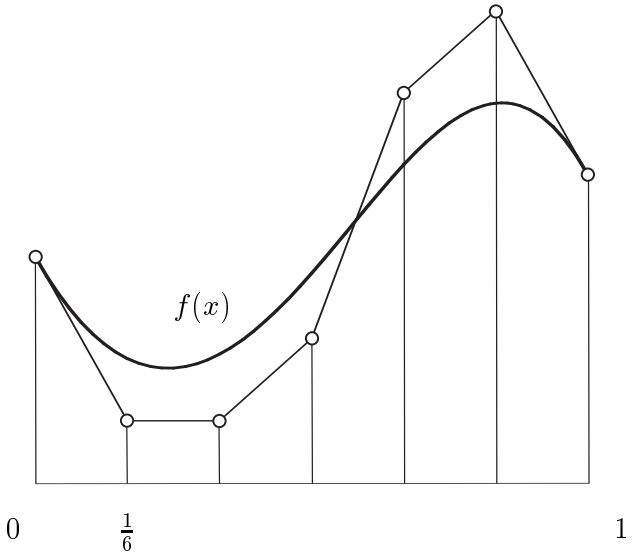


Figure 13: Nonparametric curve: polynomial $f(x)$ of degree six on $[0, 1]$ represented as a Bézier curve.

2.1.3 Nonparametric curves

The following property may seem to be trivial, but its further application (section 3) forces me to emphasize its significance. Discussing Bézier curve of degree n, both coordinates of this curve are desribed by polynomials of degree at most n (see (2) and (3)). Investigating polynomials, one can immediately see, that this group of functions is easily described by Bézier curves. Let us assume polynomial $y = f(x)$ of degree n over interval $[0, 1]$. Its parametrization as a Bézier curve is

$$\mathbf{b}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} t \\ f(t) \end{bmatrix}, \tag{17}$$

where polynomial $f(t)$ is assumed to be expressed with respect to Bernstein basis

$$f(t) = \sum_{i=0}^{n} b_i B_i^n(t). \tag{18}$$

According to the linear precision property of Bernstein polynomials, control points are uniformly distributed in the direction of x-axis (see Figure 13). Then, given polynomial is rewritten as

$$\mathbf{b}(t) = \sum_{i=0}^{n} \begin{bmatrix} i/n \\ b_i \end{bmatrix} B_i^n(t), \tag{19}$$

where real numbers b_i are known as *Bézier ordinates* and the uniform separators i/n, $i = 0, \ldots, n$ are called *Bézier abscissas*.

Of course, we are not restricted only to unit interval $[0, 1]$. Due to the affine invariance property of Bézier curves, we can describe polynomial over arbitrary domain $[a, b]$. Bézier abscissas are $a + i(b - a)/n$, $i = 0, \ldots, n$.

2.2 Bézier patches

Bézier patch of bidegree (m, n) is defined by formula

$$\mathbf{b}^{m,n}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u)B_j^n(v)\mathbf{b}_{i,j}, \quad [u,v] \in [0,1]\times[0,1], \tag{20}$$

where coefficients $\mathbf{b}_{ij}$ are called *control points* and $B_i^m(u)$, $B_j^n(v)$ are the Berstein polynomials defined in 3.

For a fixed value $u_0 \in [0,1]$, the term

$$\mathbf{b}^{n}(u_0,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u_0)B_j^n(v)\mathbf{b}_{i,j}, \quad v \in [0,1], \tag{21}$$

depends only on one parameter v and therefore defines a curve on given surface. This curve is called *isoparametric v-curve*. For all $u_0 \in [0,1]$, we obtain a system of v-curves lying on the patch. Analogously, fixing parameter v, the second system of isoparametric u-curves is obtained. The double summation in equation (20) may be easily represented in matrix form:

$$\mathbf{b}^{m,n}(u,v) = \begin{bmatrix} U_0^m(u) & \dots & U_m^m(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{00} & \dots & \mathbf{b}_{0n} \\ \vdots & & \vdots \\ \mathbf{b}_{m0} & \dots & \mathbf{b}_{mn} \end{bmatrix} \begin{bmatrix} V_0^n(v) \\ \vdots \\ V_n^n(v) \end{bmatrix}, \tag{22}$$

where the matrix $\{\mathbf{b}_{ij}\}$ is known as *geometry matrix* of the patch. Obviously, Bézier patch is uniquely defined by its control points. Similarly to the case of Bézier curves and its control polygons, the sequence of control points plays very important role. In order to avoid misinterpretation, each control point is connected to its neighborou(s) in u and v-directions by line. These lines and control points form *control points mesh*.

Example 3 Bézier patch of bidegree $(1,1)$ given by four distinct control points $\mathbf{b}_{00}$, $\mathbf{b}_{01}$, $\mathbf{b}_{10}$, $\mathbf{b}_{11}$:

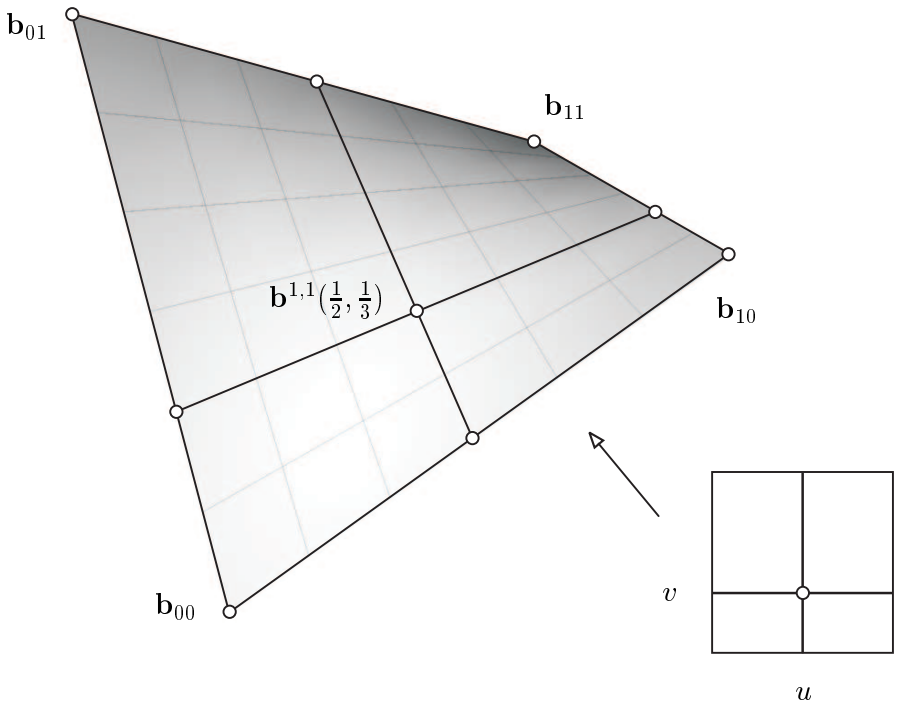


Figure 14: Bilinear Bézier patch: isoparametric curves for $u_0 = \frac{1}{2}$ and $v_0 = \frac{1}{3}$.

$$\mathbf{b}^{1,1}(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{1} B_i^m(u) B_j^n(v) \mathbf{b}_{i,j}, \quad [u,v] \in [0,1] \times [0,1] \tag{23}$$

and its matrix expression

$$\mathbf{b}^{1,1}(u,v) \;=\; \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} \\ \mathbf{b}_{10} & \mathbf{b}_{11} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}. \tag{24}$$

This bilinear Bézier patch is known as hyperbolic paraboloid, isoparametric

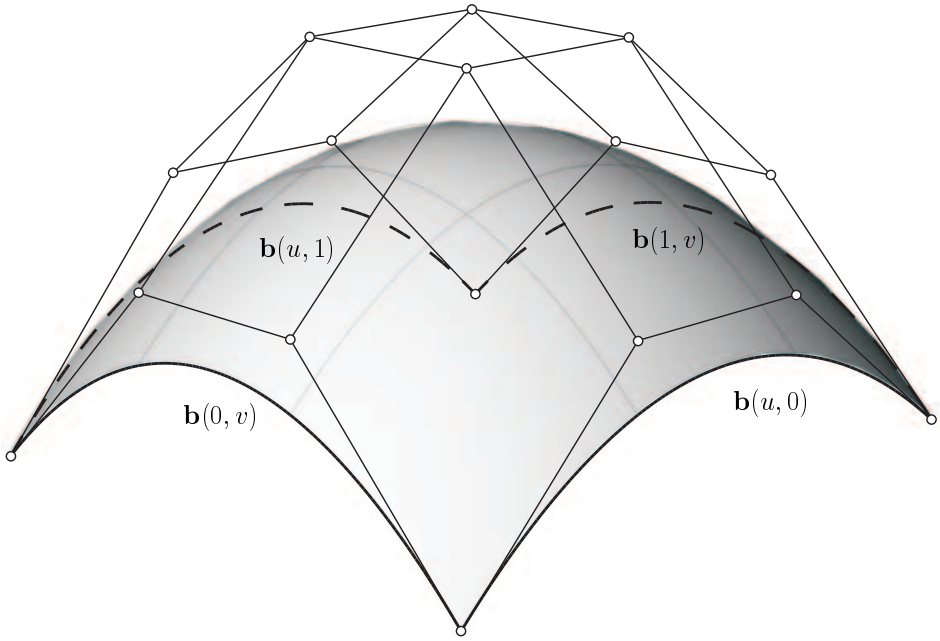


Figure 15: Bézier patch of bidegree $(3,3)$: bounding curves and control point mesh.

u- and v-curves are both lines (see Figure 14).

Properties of Bézier patches: Resulting directly from the properties of the Bernstein polynomials, we obtain

1. Coefficients of Bézier patch are barycentric coordinates, in other words:

$$\sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u)B_j^n(v) \equiv 1, \tag{25}$$

   for all u, $v \in [0,1]$;

2. affine invariance: computing the affine image of control points mesh

of the surface and control points mesh of the image of the surface, we obtain the same result;

3. convex hull property: $B_i^m(u)$, $B_j^n(v)$ are nonnegative for all $0 \leq u, v \leq 1$. With respect to equations (25) and (20), $\mathbf{b}^{m,n}(u,v)$ lies inside the convex hull of control points mesh;

4. boundary curves: Evaluating the patch $\mathbf{b}^{m,n}(u,v)$ for $u = 0,1$, $v = 0,1$, four Bézier curves $\mathbf{b}(0,v)$, $\mathbf{b}(1,v)$, $\mathbf{b}(u,0)$, $\mathbf{b}(u,1)$ are obtained, respectively. Control polygons are formed by appropriate boundary control points (see Fig. 15);

5. tangent planes in the corners: computing $\frac{\partial}{\partial u}\mathbf{b}^{m,n}(u,v)$, $\frac{\partial}{\partial v}\mathbf{b}^{m,n}(u,v)$ and evaluating them for $u = 0,1$, $v = 0,1$, the corner tangent planes are defined by three corner control points (see Figure 16).

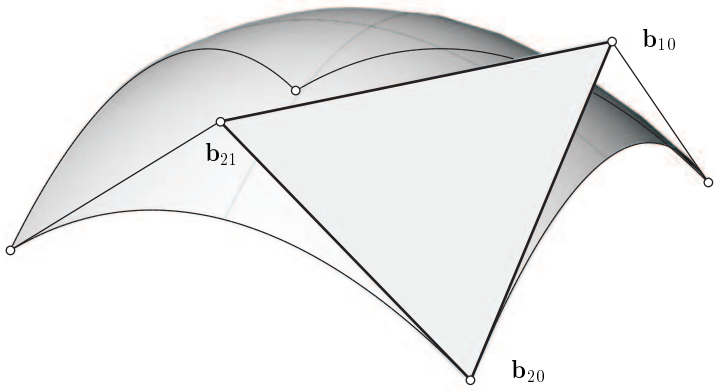


Figure 16: Biquadratic Bézier patch: tangent plane in the corner control point.

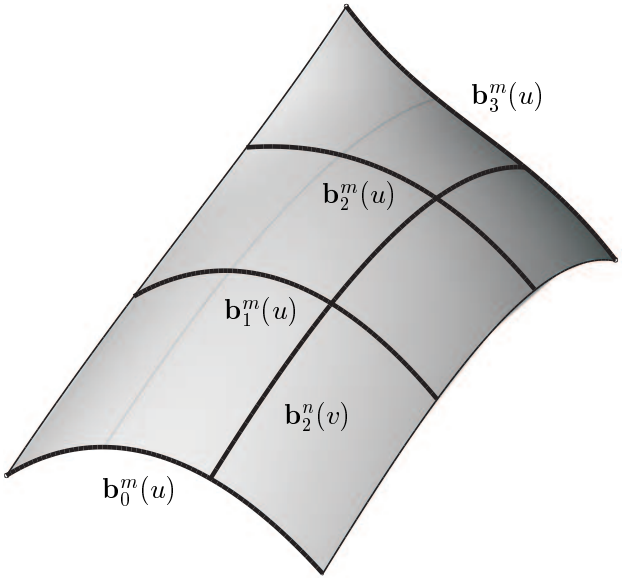


Figure 17: Tensor product surface: shaping u-curve $\mathbf{b}^m(u)$ sweeps along the path v-curve $\mathbf{b}_2^n(v)$.

2.2.1 Tensor product

Definition of Bézier patch via control points mesh (equation (20)) is classical approach to this topic. Observing this equation thoroughly, one can interprete this definition more from the kinematic point of view. Fixing the value of parameter $v = 0$, we obtain a starting curve

$$\mathbf{b}_0^m(u) = \mathbf{b}^{m,n}(u,0) = \sum_{i=0}^{m} B_i^m(u)\mathbf{b}_{i,0}, \quad u \in [0,1]. \tag{26}$$

Varying the parameter v, the system Φ of Bézier curves of degree m is obtained. This one-parametrical system Φ may be apprehended as a trajectory of curve $\mathbf{b}_0^m(u)$ during certain affine kinematic motion. In other words: curve

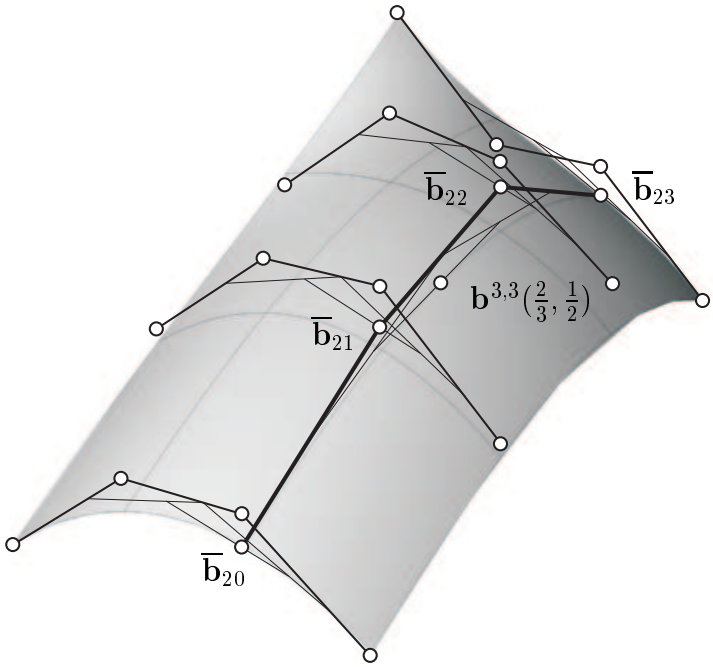


Figure 18: Tensor product bicubic patch: construction of a point $\mathbf{b}^{3,3}(\frac{2}{3},\frac{1}{2})$ via five de Casteljau algorithms.

$\mathbf{b}_0^m(u)$ is swept out, changing continuously its shape (see Fig. 17). Clearly, each point travels on the Bézier curve of degree n. Then, u-curves are perceived as shaping curves, v-curves represent trajectories of points $\mathbf{b}_0^m(u)$ during this motion. Resulting from the symmetry of equation (20), u and v-curves may be replaced by each other to receive the same surface. Then, surface $\mathbf{b}^{m,n}(u,v)$ is known as *tensor product surface*.

Consequently, the construction of a point on a tensor product surface may be easily reduced to several one-dimensional steps. Figure 18 shows the application of de Casteljau algorithm for u-curves. The control polygon for isoparametric v-curve is acquired.

2.2.2 Degree elevation

Following the tensor product approach to Bézier patches, the degree elevation process may be easily reduced to several univariate degree elevation steps of Bézier curves (see section 2.1.2). Let us assume Bézier patch of bidegree (m, n) as a one of bidegree $(m+1, n)$. The goal is to find coefficients $\mathbf{b}_{i,j}^{(1,0)}$ such that

$$\mathbf{b}^{m,n}(u,v) = \sum_{j=0}^{n} \underbrace{\sum_{i=0}^{m+1} U_i^{m+1}(u)\mathbf{b}_{i,j}^{1,0}}_{q_i(u)} B_j^n(v). \tag{27}$$

The $n+1$ terms $q_i(u)$ express $n+1$ univariate degree elevation, that was discussed in section 2.1.2. Applying repeatedly (15), the coefficinets $\mathbf{b}_{i,j}^{(1,0)}$ are directly obtained:

$$\mathbf{b}_{i,j}^{(1,0)} = \frac{i}{m+1}\mathbf{b}_{i-1,j} + (1 - \frac{i}{m+1})\mathbf{b}_{i,j}, \tag{28}$$

where $i = 0, \ldots, m+1$ and $j = 0, \ldots, n$. Interpreting this result, control point mesh of the degree elevated Bézier patch is created from the original one by $n+1$ elevations of row control points of isoparametric u-curves (see Fig. 19).

Requesting the progressive degree elevation by k degrees in one direction, the formula (16) is applied. The degree elevation in the v-direction is defined analougously. To receive the degree elevated surface by one in both directions, we elevate in u-direction and the v-direction. Due to the tensor product properties, it is irrelevant whether we elevate in u-direction first and then in v-direction or vice versa.

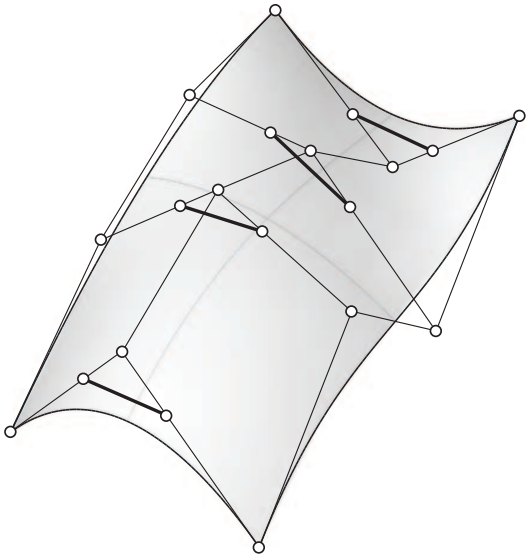


Figure 19: Degree elevation of Bézier patch: reduction to several univariate degree elevation steps.

2.2.3 Nonparametric patches

Analogously to the section (2.1.3), one of the most significant applications of Bézier patches is related to polynomials. The graph of polynomial f in two variables x and y is easy expressed as a Bézier patch by parametrization

$$\mathbf{P}(x,y) = \begin{bmatrix} x \\ y \\ f(x,y) \end{bmatrix}, \tag{29}$$

where polynomial f is assumed to be expressed in Bernstein form

$$f(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(x) B_j^n(y) p_{i,j}, \tag{30}$$

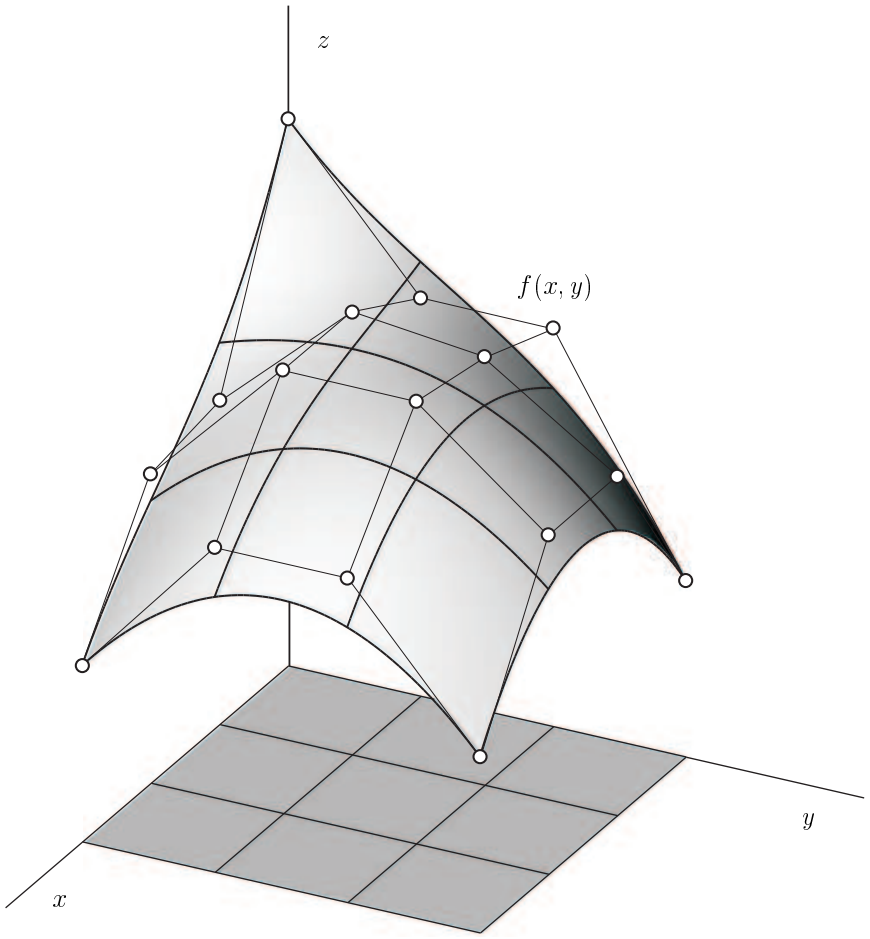


Figure 20: Nonparametric patch: polynomial $f(x, y)$ represented as Bézier patch.

over unit square $[0, 1] \times [0, 1]$. Due to linear precision property of Bernstein polynomials, the control point mesh is given by

$$\mathbf{b}_{i,j}(x, y) = \begin{bmatrix} i/m \\ j/n \\ b_{i,j} \end{bmatrix}. \tag{31}$$

The orthogonal projection of $\mathbf{b}_{i,j}$ into the plane $z = 0$ create a uniform point mesh known as *Bézier abscissas* and coeffients $b_{i,j}$ are called *Bézier ordinates*. This is ilustrated in Figure 20. Of course, we are not restricted only to unit domain $[0, 1] \times [0, 1]$. Using affine transformation, we can map unit domain into arbitrary domain $[a, b] \times [c, d]$ to obtain Bézier patch over requested domain.

2.3 Linear space of polynomials

Investigating polynomials and its degree reduction, it is essential to mention several notes concernig the linear space of polynomials focusing on its norms.

Let Π^n be the linear space of polynomials of degree at most n, with the basis $(B_i^n)_{i=0,\dots,n}$, where

$$B_i^n(t) = \binom{n}{i} \frac{(t-\alpha)^i (\beta - t)^{n-i}}{(\beta - \alpha)^n} \tag{32}$$

are the Bernstein polynomials with respect to a certain interval $[\alpha, \beta] \subset \mathbb{R}$.

Let us define L^2 inner product

$$\langle f, g \rangle^{[\alpha,\beta]} = \int_\alpha^\beta f(t)\, g(t)\, \mathrm{d}t \tag{33}$$

with respect to the interval $[\alpha, \beta]$ and the norm

$$\|f\|_2^{[\alpha,\beta]} = \frac{1}{h} \sqrt{\langle f, f \rangle^{[\alpha,\beta]}}, \tag{34}$$

where $h = \beta - \alpha$, induced by it.

In this definition of the norm, the factor $1/h$ is introduced in order to obtain a norm which is *invariant under affine transformations* of the t–axis (see

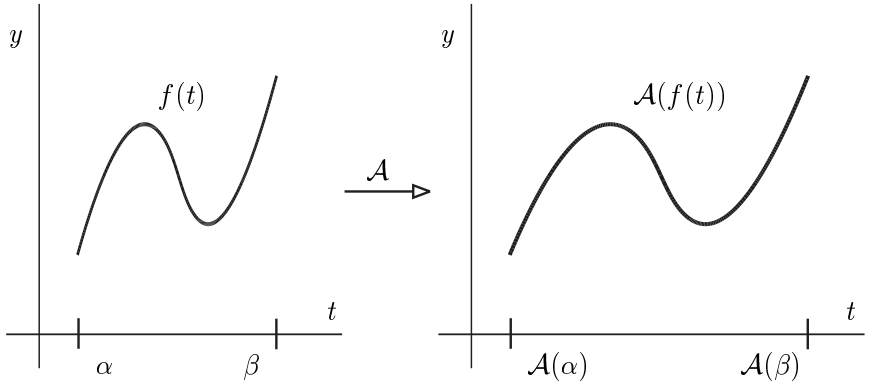


Figure 21: The affine transformation: the L^2 norm of f and its affine image $\mathcal{A}(f)$ are the same.

Figure 21). More precisely, for any affine transformation

$$\mathcal{A} : t \mapsto A_0 + A_1\, t \tag{35}$$

with $A_1 \neq 0$, the norms of f with respect to the interval $[\alpha, \beta]$ and of $f \circ \mathcal{A}^{-1}$ with respect to the interval $\mathcal{A}([\alpha, \beta])$ are identical,

$$\|f\|_2^{[\alpha,\beta]} = \|f \circ \mathcal{A}^{-1}\|_2^{\mathcal{A}([\alpha,\beta])}. \tag{36}$$

Various norms on Π^n are available. Focusing on polynomials from the point of view of Bézier curves, we define the maximum norm on BB-coefficients of polynomial f:

$$\|f\|_{\mathrm{BB},\infty}^{[\alpha,\beta]} = \max_{i=0,\ldots,n} |b_i|, \tag{37}$$

where constants b_i are the y-coordinates of control points (compare with 18 and see Figure 22).

The maximum norm of polynomial f is given by

$$\|f\|_\infty^{[\alpha,\beta]} = \max_{t\in[\alpha,\beta]} |f(t)|. \tag{38}$$

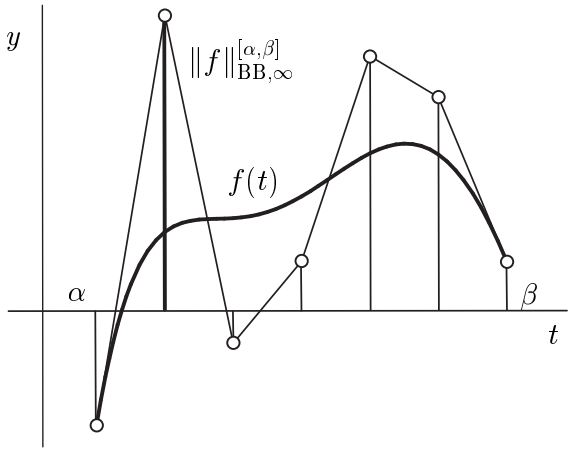


Figure 22: Maximum norm on BB-coefficients: polynomial f and its norm $\|f\|_{\mathrm{BB},\infty}^{[\alpha,\beta]}$ with respect to the interval $[\alpha, \beta]$.

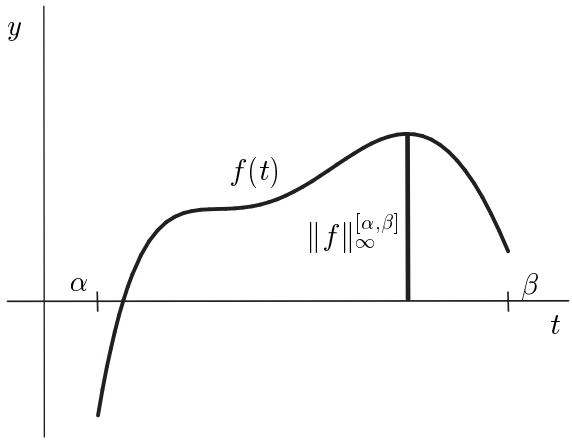


Figure 23: Maximum norm: polynomial f and its norm $\|f\|_{\infty}^{[\alpha,\beta]}$ with respect to the interval $[\alpha, \beta]$.

This is ilustrated in Figure 23.

One can easily check that all three norms – L^2 norm, maximum norm and maximum norm on BB-coefficients – satisfy the definition of the norm. Its affine invariance is obvious.

2.4 Degree reduction and dual basis

The process of approximating a polynomial of degree n by a polynomial of degree k, where $k < n$, with respect to a suitable norm, is called *degree reduction*. We consider the spaces Π^n and $\Pi^k \subset \Pi^n$, along with the L^2 norm defined in section 2.3.

Applying degree reduction with respect to this norm to the given polynomial p gives the unique polynomial $q \in \Pi^k$ which minimizes $\|p - q\|_2^{[\alpha,\beta]}$, i.e.,

$$q = \arg\min_{q \in \Pi^k} \|p - q\|_2^{[\alpha,\beta]}. \tag{39}$$

Various techniques for computing q are available (see introduction for references). We describe a simple technique which is based on the dual basis of the Bernstein polynomials.

The *dual basis* to the Bernstein basis of Π^k consists of the unique polynomials D_j^k of degree k which satisfy

$$\langle B_i^k, D_j^k \rangle^{[\alpha,\beta]} = \delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}, \quad i, j = 0 \ldots k, \tag{40}$$

see Figure 25. The polynomials D_j^k can be represented with respect to the Bernstein basis,

$$D_i^k(t) = \frac{1}{h} \sum_{j=0}^{k} c_{i,j}\, B_j^k(t), \quad i = 0, \ldots, k, \tag{41}$$

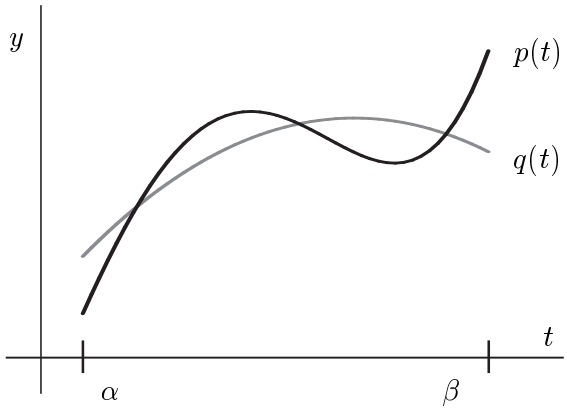


Figure 24: Degree reduction: polynomial p and its best approximant q with respect to L^2 norm.

with the coefficients

$$c_{p,q} = \frac{(-1)^{p+q}}{\binom{k}{p}\binom{k}{q}} \sum_{j=0}^{\min(p,q)} (2j+1)\binom{k+j+1}{k-p}\binom{k-j}{k-p}\binom{k+j+1}{k-q}\binom{k-j}{k-q} \tag{42}$$

which have been derived in Jüttler (1998), and $h = \beta - \alpha$. Alternatively, these polynomials can be computed using a recurrence relation involving dual basis polynomials of lower degree and Legendre polynomials Ciesielski (1987).

The polynomial q obtained by applying degree reduction to p (see (80) and (39)) with respect to the interval $[\alpha, \beta]$ may be computed from

$$q(t) = \sum_{j=0}^{k} \langle p(t), D_j^k(t) \rangle^{[\alpha,\beta]} B_j^k(t) = \sum_{j=0}^{k} \left(\sum_{i=0}^{n} b_i \beta_{i,j}^{n,k} \right) B_j^k(t), \tag{43}$$

with the coefficients

$$\beta_{i,j}^{n,k} = \langle B_i^n(t), D_j^k(t) \rangle^{[\alpha,\beta]}. \tag{44}$$

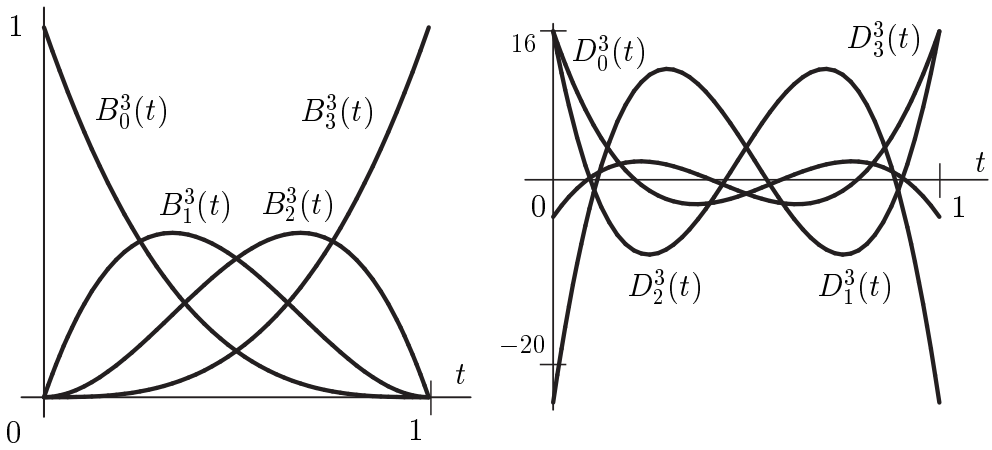


Figure 25: Dual basis: The Bernstein basis of degree 3 and the associated dual basis functions.

Using the identity

$$\langle B_i^m, B_j^n \rangle^{[\alpha,\beta]} = h \frac{\binom{m}{i}\binom{n}{j}}{(m+n+1)\binom{m+n}{i+j}}, \tag{45}$$

these coefficients can be computed from (86) and (42). Note that these coefficients do not depend on the interval $[\alpha, \beta]$, since the factors h in (86) and (45) cancel each other.

Example 4 The degree reduction coefficients for $n = 5$ and $k = 2$ form the matrix

$$(\beta_{i,j}^{5,2})_{i=0,..,5;j=0,...,2} = \begin{bmatrix} \frac{23}{28} & -\frac{3}{7} & \frac{3}{28} \\ \frac{9}{28} & \frac{2}{7} & -\frac{3}{28} \\ 0 & \frac{9}{14} & -\frac{1}{7} \\ -\frac{1}{7} & \frac{9}{14} & 0 \\ -\frac{3}{28} & \frac{2}{7} & \frac{9}{28} \\ \frac{3}{28} & -\frac{3}{7} & \frac{23}{28} \end{bmatrix}. \tag{46}$$

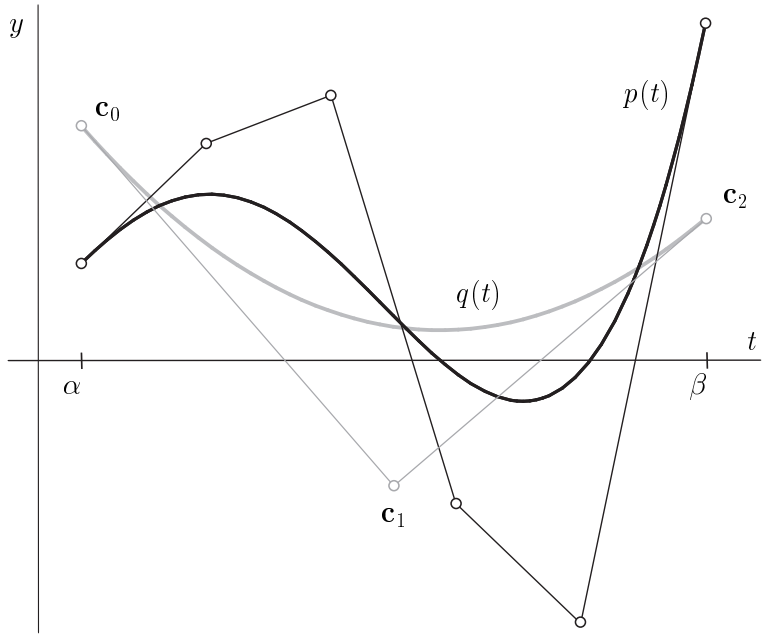


Figure 26: The best quadratic approximant: given polynomial p and its best approximant q with respect to L^2 norm; y-coordinates of control points $\mathbf{c}_0$, $\mathbf{c}_1$, $\mathbf{c}_2$ are computed via matrix (46); x-coordinates are distributed uniformly on $[\alpha, \beta]$.

The coefficients vector (c_0, c_1, c_2) of q is obtained by multiplying the row vector $(b_0, \ldots, b_5)$ of the coefficients of p by this matrix. Representing parabola q by Bézier curve, the control points are

$$\mathbf{c}_0 = \begin{bmatrix} \alpha \\ c_0 \end{bmatrix}, \qquad \mathbf{c}_1 = \begin{bmatrix} (\alpha + \beta)/2 \\ c_1 \end{bmatrix}, \qquad \mathbf{c}_2 = \begin{bmatrix} \beta \\ c_2 \end{bmatrix}. \tag{47}$$

This illustrates Figure 26.

Algorithm 1 `bezclip` $(p, [\alpha, \beta])$ {Bézier clipping}

1: **if** length of interval $[\alpha, \beta] \geq \varepsilon$ **then**
2: $\mathcal{C} \leftarrow$ convex hull of the control points of p with respect to $[\alpha, \beta]$.
3: **if** $\mathcal{C}$ intersects t-axis **then**
4: Find $[\alpha', \beta']$ by intersecting $\mathcal{C}$ with the t–axis.
5: **if** $\mid \alpha' - \beta' \mid < \frac{1}{2} \mid \alpha - \beta \mid$ **then**
6: **return** (`bezclip` $(p, [\alpha', \beta'])$)
7: **else**
8: **return** (`bezclip` $(p, [\alpha, \frac{1}{2}(\alpha + \beta)])$ $\cup$ `bezclip` $(p, [\frac{1}{2}(\alpha + \beta), \beta])$).
9: **end if**
10: **else**
11: **return** $(\emptyset)$
12: **end if**
13: **else**
14: **return** $([\alpha, \beta])$
15: **end if**

2.5 Bézier clipping and its convergence rate

Bézier clipping, presented in Nishita, Sederberg and Kakimoto (1990), is robust polynomial solver, that gives *all* roots of given polynomial p on given interval $[\alpha, \beta]$. Presenting a new polynomial solver with similar structure and comparing it with Bézier clipping (section 3), we recall this method at first.

Bézier clipping, see Algorithm 1 (`bezclip`), uses the convex hull property of Bernstein–Bézier representations in order to generate one or more intervals of maximum length ε which contain(s) the roots.

The polynomial p is represented by its Bézier coefficients with respect to the current interval $[\alpha, \beta]$. The *graph* of p can be described as a *parametric* Bézier

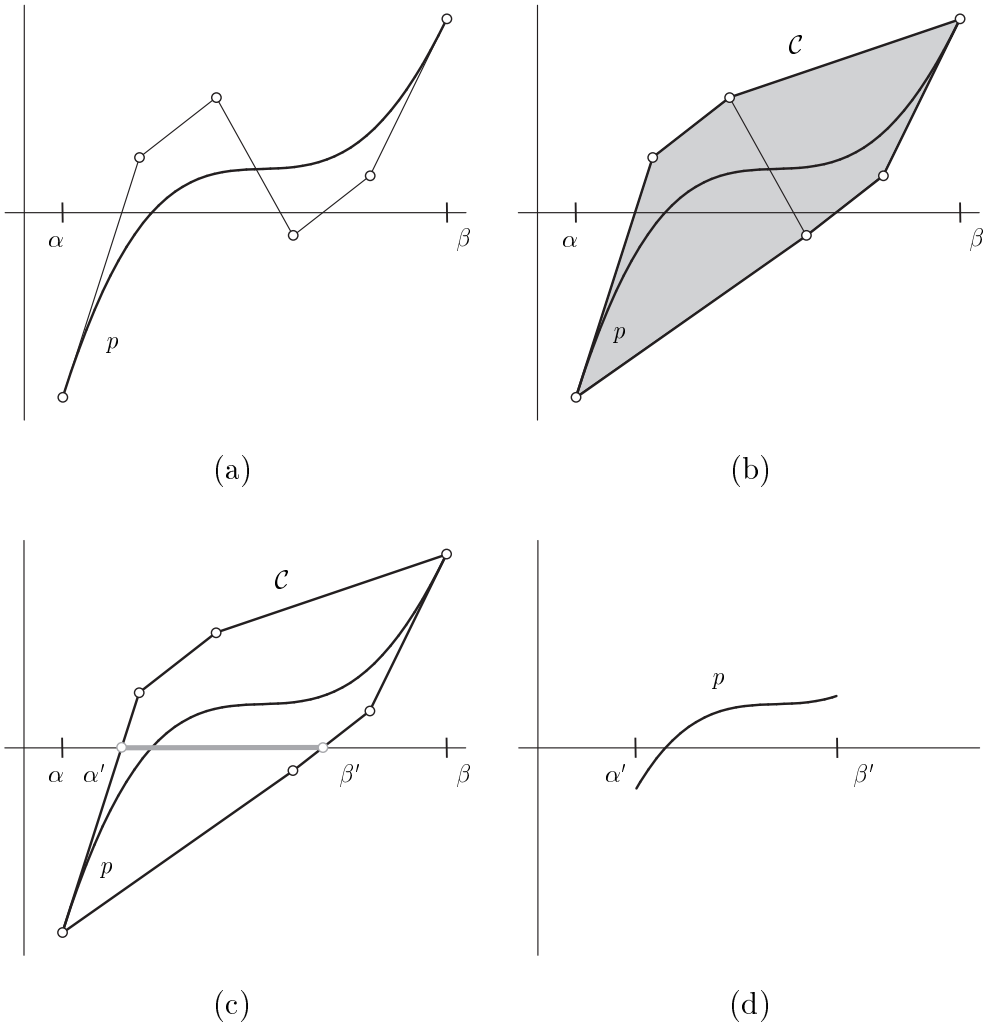


Figure 27: One iteration of `bezclip`: (a) The polynomial p is represented in BB-form on $[\alpha, \beta]$, (b) The convex hull $\mathcal{C}$ of control polygon is constructed, (c) $\mathcal{C}$ is intersected with t–axis in order to define new interval $[\alpha', \beta']$, (d) p is subdivided on $[\alpha', \beta']$.

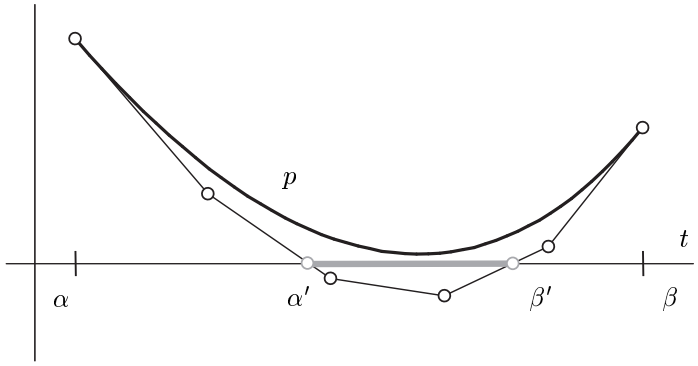


Figure 28: False positive answer: If the lenght of the interval $[\alpha', \beta']$ is less then prescribed accuracy ϵ, `bezclip` returns $[\alpha', \beta']$ as root-containing interval.

curve (see section 2.1.3) with control points

$$\mathbf{b}_i = (\frac{(n-i)\alpha + i\beta}{n}, b_i), \quad i = 0, \ldots, n. \tag{48}$$

Due to the convex–hull property, the graph lies within the convex hull $\mathcal{C}$ of the control points $(\mathbf{b}_i)_{i=0,\ldots,n}$. Consequently, all roots of the polynomial p are contained in the interval which is obtained by intersecting $\mathcal{C}$ with the t–axis. This observation, which is illustrated by Figure 27c, is used in lines 2–4 of the algorithm to generate the next interval.

In line 6, the de Casteljau algorithm is applied twice to generate the coefficients with respect to the subinterval $[\alpha', \beta']$. Similar, it is applied once in line 8, in order to bisect the interval.

For any root contained in $[0, 1]$, the call `bezclip`$(p, [0, 1])$ returns an interval containing that root. Bézier clipping may produce false positive answers (i.e., intervals not containing any root) if the graph of the polynomial gets very close to the t–axis.

In order to study the efficiency of Bézier clipping, we analyze the sequence $(h_i)_{i=0}^{\infty}$ of the lengths of the intervals $[\alpha, \beta]$ generated after calling `bezclip` i times. Note that algorithm `bezclip` acts recursively, and combines bisection with clipping steps. Here we follow only one path in the execution tree which leads towards one of the roots. As observed by Nishita and Sederberg (1990), this sequence has convergence rate 2, provided that it leads to a single root. In the case of multiple roots, however, only *linear* convergence is achieved. (See Gautschi (1997) for more information about convergence rates).

3 Computing roots via quadratic clipping

In this section, we describe a new algorithm `quadclip` for isolating the roots of univariate polynomial and analyze its convergence rates in the cases of roots with multiplicities 1 and 2. Then, we present a detailed comparison with the standard technique of Bézier clipping.

3.1 The root–finding problem

Let Π^n be the linear space of polynomials of degree n, with the basis $(B_i^n)_{i=0,\dots,n}$, where

$$B_i^n(t) = \binom{n}{i} \frac{(t-\alpha)^i(\beta-t)^{n-i}}{(\beta-\alpha)^n} \tag{49}$$

are the Bernstein polynomials with respect to a certain interval $[\alpha, \beta] \subset \mathbb{R}$. Any polynomial $p \in \Pi^n$ can be described by its Bernstein–Bézier representation with respect to that interval,

$$p(t) = \sum_{i=0}^{n} b_i B_i^n(t), \quad t \in [\alpha, \beta], \tag{50}$$

with certain Bernstein–Bézier (BB) coefficients $b_i \in \mathbb{R}$.

We consider a given polynomial $p \in \Pi^n$ in Bernstein–Bézier representation with respect to the interval $[\alpha, \beta]$. All roots of p within $[\alpha, \beta]$ are to be found. More precisely, we want to generate a set of intervals of maximum length ε which contain the roots, where the parameter ε specifies the desired accuracy.

3.2 Algorithm

Based on degree reduction to a quadratic polynomial (see section 2.4), we propose a new technique for computing the roots, see Algorithm 3 (`quadclip`).

Some steps of the algorithm will be explained in more detail:

- In line 2 of the algorithm, we generate the best quadratic approximant q with respect to the L^2 norm on the current interval $[\alpha, \beta]$, see Fig. 29(b). This is achieved by multiplying the row vector of Bézier coefficients of p with the degree reduction matrix $(\beta_{i,j}^{n,2})_{i=0,\ldots,n;j=0,1,2}$. These coefficients are precomputed and stored in a lookup–table.

- In order to obtain the bound δ on

$$\|p - q\|_\infty^{[\alpha,\beta]} = \max_{t\in[\alpha,\beta]} |p(t) - q(t)|, \tag{51}$$

  see line 3, we raise the degree of the Bernstein–Bézier representation of the quadratic polynomial q to n. Similar to degree reduction, this is achieved by multiplying the row vector of Bézier coefficients of q with the degree raising matrix $(\beta_{i,j}^{2,n})_{i=0,2,1;j=0,\ldots,n}$. These coefficients are again precomputed and stored in a lookup–table, see Example 5. The bound is chosen as

$$\delta = \max_{i=0,\ldots,n} |b_i - c_i|, \tag{52}$$

  see Fig. 29(c), where b_i and c_i are the coefficients of the Bernstein–Bézier representations of p and q of degree n with respect to $[\alpha, \beta]$, respectively.

Algorithm 2 `quadclip`$(p, [\alpha, \beta])$ {Quadratic clipping}

if length of interval $[\alpha, \beta] \geq \varepsilon$ **then**
  $q \leftarrow$ generate a quadratic polynomial by applying degree reduction with respect to the L^2 inner product on $[\alpha, \beta]$ to p.
  $\delta \leftarrow$ compute bound on $\|p - q\|_\infty^{[\alpha,\beta]}$ by comparing the Bernstein–Bézier representations of p and q.
  $m \leftarrow q - \delta$ {lower bound}
  $M \leftarrow q + \delta$ {upper bound}
  if the strip enclosed by m, M does not intersect the t–axis within $[\alpha, \beta]$ **then**
    return $(\emptyset)$
  else
    Find intervals $[\alpha_i, \beta_i]$, $i = 1, \ldots, k$, by intersecting m, M with the t–axis. The number k of intervals is either 1 of 2.
    if $\max_{i=1,\ldots,k} |\alpha_i - \beta_i| > \frac{1}{2}|\alpha - \beta|$ **then**
      return (`quadclip` $(p,[\alpha, \frac{1}{2}(\alpha+\beta)])$ $\cup$ `quadclip` $(p,[\frac{1}{2}(\alpha+\beta), \beta]))$.
    else
      $S \leftarrow \emptyset$
      for $i = 1, \ldots, k$ **do**
        $S \leftarrow S \cup$ `quadclip`$(p, [\alpha_i, \beta_i])$
      end for
      return (S)
    end if
  end if
else
  return $([\alpha, \beta])$
end if

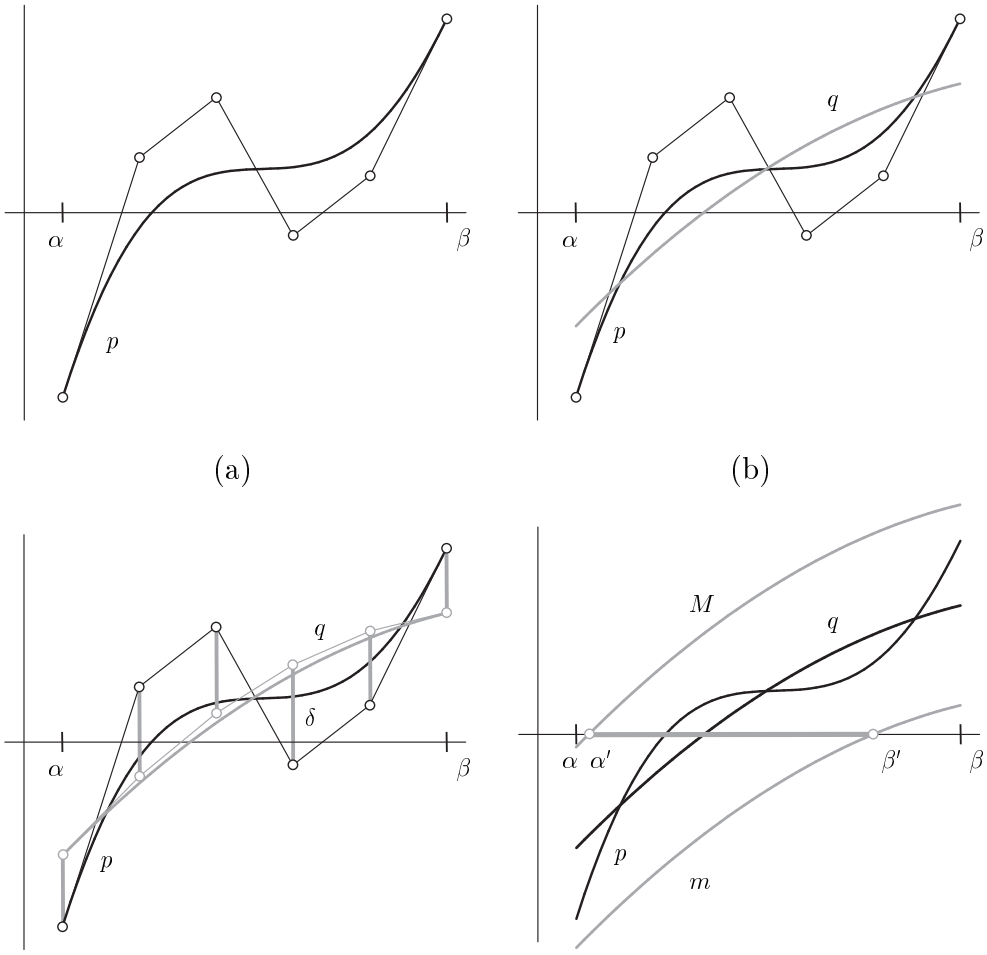


Figure 29: One iteration of `quadclip`: (a) the polynomial p is represented in BB-form on $[\alpha, \beta]$, (b) q – the best quadratic approximant of p with respect to L_2 norm, (c) the error bound δ is obtained as the maximum length of the thick grey vertical bars, (d) the lower and upper bounds $m = q - \delta$ and $M = q + \delta$, the intersection of the strip enclosed by them with the t–axis defines the new interval $[\alpha', \beta']$.

- In lines 4 and 5, the bound δ is used to construct quadratic polynomials m and M satisfying

$$\forall t \in [\alpha, \beta]: \quad m(t) \leq p(t) \leq M(t). \tag{53}$$

- In lines 6–19 we analyze the strip enclosed by m and M and its intersection with the t–axis, see Fig. 30. If the intersection is empty, then no roots exist. Otherwise, the intersection consists of either one or two intervals that contain the roots. Their boundaries are found by solving two quadratic equations, see Remark 2.

- If the length(s) of this/these interval(s) is/are sufficiently small, when compared to the length of the previous interval $[\alpha, \beta]$, then `quadclip` is applied to them (lines 14–16). Otherwise we bisect the interval $[\alpha, \beta]$ and apply `quadclip` to the two halves (line 11).

For any root contained in $[\alpha, \beta]$, the call `quadclip`$(p, [\alpha, \beta])$ returns an interval containing that root. Similar to Bézier clipping, quadratic clipping may produce false positive answers (i.e., intervals not containing any root) if the graph of the polynomial gets very close to the t–axis.

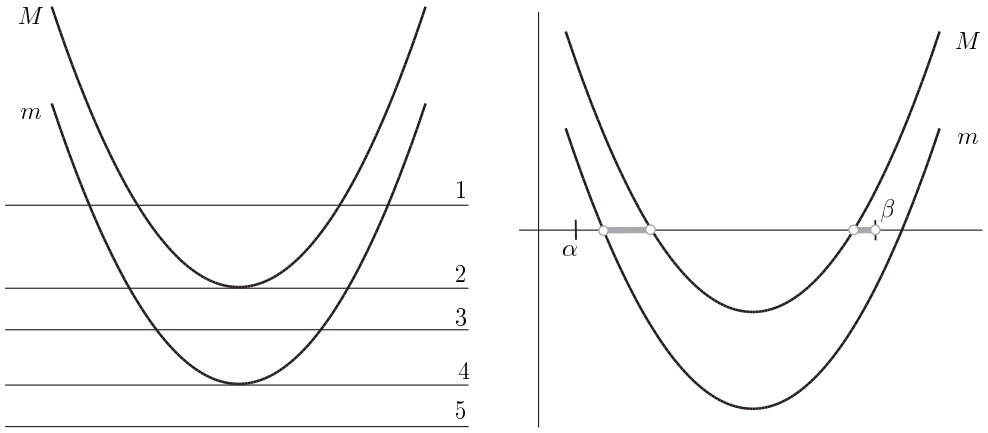
Example 5 The degree raising coefficients for $n = 5$ and $k = 2$ form the matrix

$$(\beta_{i,j}^{2,5})_{i=1,\ldots,2;j=0,\ldots,5} = \begin{bmatrix} 1 & \frac{3}{5} & \frac{3}{10} & \frac{1}{10} & 0 & 0 \\ 0 & \frac{2}{5} & \frac{3}{5} & \frac{3}{5} & \frac{2}{5} & 0 \\ 0 & 0 & \frac{1}{10} & \frac{3}{10} & \frac{3}{5} & 1 \end{bmatrix}, \tag{54}$$

where coefficients are computed via dual basis

$$\beta_{i,j}^{2,5} = \langle B_i^2(t), D_j^5(t) \rangle^{[\alpha,\beta]}, \tag{55}$$

(see section 2.4). The vector of BB-representation of degree-raised parabola q is obtained by multiplying the row vector $(\overline{c}_0, \overline{c}_1, \overline{c}_2)$ by matrix $(\beta_{i,j}^{2,5})$, see Figure 31.



(a) (b)

Figure 30: The definition of a new domain: a) The parabolic strip – enclosed by M and m – is intersected with t-axis, five options may occur, b) Case 1, intervals defined by roots of M and m are intersected with the original interval $[\alpha, \beta]$ in order to obtain the new domains.

Remark 1 In order to construct bound δ (see eq. 52)), the maximum norm on vectors of BB-representation of p and q is used. Clearly,

$$\|p-q\|_{\infty}^{[\alpha,\beta]} = \sum_{i=0}^{n} |b_i - c_i| B_i^n(t) \leq \sum_{i=0}^{n} \max_{i=0,\ldots,n} |b_i - c_i| B_i^n(t) =$$

$$= \max_{i=0,\ldots,n} |b_i - c_i| = \|p-q\|_{\mathrm{BB},\infty}^{[\alpha,\beta]} = \delta, \tag{57}$$

therefore the bounds M and m are well defined.

Remark 2 The roots of a quadratic polynomial (cf. lines 6 and 9 of the algorithm) $g(t) = B_0^2(t)\, d_0 + B_1^2(t)\, d_1 + B_2^2(t)\, d_2$ are $t_{1|2} = (1 - \tau_{1|2})\alpha + \tau_{1|2}\beta$ where

$$\tau_{1|2} = \frac{d_1 - d_0 \pm \sqrt{D}}{d_2 - 2d_1 + d_0}. \tag{58}$$

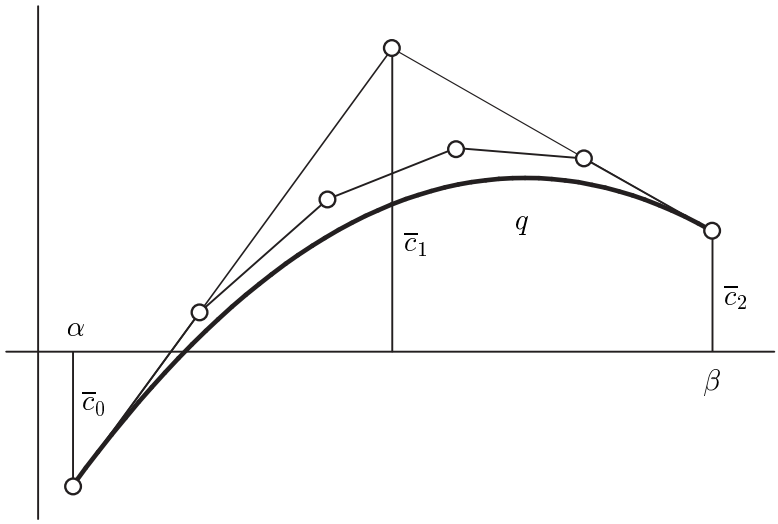


Figure 31: The degree-raised parabola: The best quadratic approximant q, its original BB-representation $(\overline{c}_0, \overline{c}_1, \overline{c}_2)$ and the control polygon of q after degree raising.

with $D = d_1^2 - d_0\, d_2$. If $|d_2 - 2d_1 + d_0|$ is below a user–defined threshold (which depends on the accuracy of the numerical computation), then the computation of the roots via (58) becomes numerically unstable. In this situation we apply Bézier clipping to the control polygon of g in order to bound the roots.

3.3 Convergence rate

In order to make this thesis self–contained, we start this section by formulating two technical lemmas.

Lemma 1 *For any given polynomial p, there exists a constant C_p depending*

solely on p, such that for all intervals $[\alpha, \beta] \subseteq [0, 1]$ *the bound* δ *generated in line 3 of Algorithm* `quadclip` *satisfies* $\delta \leq C_p\, h^3$*, where* $h = \beta - \alpha$*.*

Proof 1 Due to the equivalence of norms in finite–dimensional real linear spaces, there exist constants C_1 and C_2 such that

$$\forall r \in \Pi^n : \quad \|r\|_{\mathrm{BB},\infty}^{[\alpha,\beta]} \leq C_1 \|r\|_2^{[\alpha,\beta]} \quad \text{and} \quad \|r\|_2^{[\alpha,\beta]} \leq C_2 \|r\|_\infty^{[\alpha,\beta]}, \tag{59}$$

where the three norms are the maximum (ℓ_∞) norm of the Bernstein–Bézier coefficients, the L^2 norm and the maximum norm (see section 2.3) all with respect to the interval $[\alpha, \beta]$. The constants C_1 and C_2 do not depend on the

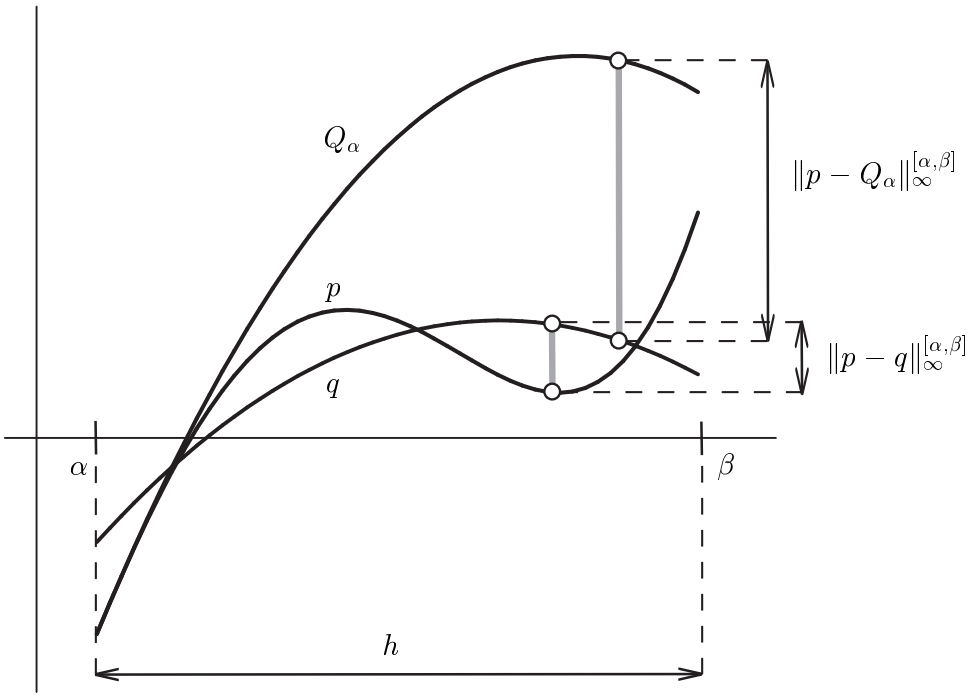


Figure 32: The lenghts of both thick grey vertical bars are of $O(h^3)$.

given interval $[\alpha, \beta]$, since all three norms are invariant with respect to affine transformations of the t–axis; cf. (83) and (84).

Consequently,

$$\begin{aligned} \delta = \|p-q\|_{\mathrm{BB},\infty}^{[\alpha,\beta]} \le C_1 \|p-q\|_2^{[\alpha,\beta]} \le C_1 \|p-Q_\alpha\|_2^{[\alpha,\beta]} \le \\ \le C_1 C_2 \|p-Q_\alpha\|_\infty^{[\alpha,\beta]} \le \frac{1}{6} C_1 C_2 \max_{t_0 \in [0,1]} |p'''(t_0)| \, h^3, \end{aligned} \tag{60}$$

where Q_α is the quadratic Taylor polynomial at $t = \alpha$ to p and p''' is the third derivative. □

Lemma 2 *For any given polynomial p there exist constants V_p, D_p and A_p depending solely on p, such that for all intervals $[\alpha, \beta] \subseteq [0, 1]$ the quadratic polynomial q obtained by applying degree reduction to p satisfies*

$$\|p-q\|_\infty^{[\alpha,\beta]} \le V_p \, h^3, \;\; \|p'-q'\|_\infty^{[\alpha,\beta]} \le D_p \, h^2, \;\; \textit{and} \;\; \|p''-q''\|_\infty^{[\alpha,\beta]} \le A_p \, h, \tag{61}$$

with $h = \beta - \alpha$, where $\|.\|_\infty^{[\alpha,\beta]}$ is defined as in (**??**).

Proof 2 Similar to the proof of the previous lemma, it can be shown that the norm

$$\|r\|_\star^{[\alpha,\beta]} = \|r\|_\infty^{[\alpha,\beta]} + h \, \|r'\|_\infty^{[\alpha,\beta]} + h^2 \, \|r''\|_\infty^{[\alpha,\beta]}, \tag{62}$$

satisfies

$$\|r\|_\star^{[\alpha,\beta]} \le C_3 \|r\|_2^{[\alpha,\beta]} \tag{63}$$

where the constant C_3 does not depend on the interval $[\alpha, \beta]$, again due to the affine invariance. Therefore, and using similar arguments as in the previous proof,

$$\begin{aligned} \|p-q\|_\star^{[\alpha,\beta]} = \|p-q\|_\infty^{[\alpha,\beta]} + h \, \|p'-q'\|_\infty^{[\alpha,\beta]} + h^2 \, \|p''-q''\|_\infty^{[\alpha,\beta]} \le \\ \le C_3 \|p-q\|_2^{[\alpha,\beta]} \le C_3 \|p-Q_\alpha\|_2^{[\alpha,\beta]} \le C_2 C_3 \|p-Q_\alpha\|_\infty^{[\alpha,\beta]} \le \\ \le \frac{1}{6} C_2 C_3 \max_{t_0 \in [0,1]} |p'''(t_0)| \, h^3, \end{aligned} \tag{64}$$

where Q_α is the quadratic Taylor polynomial at $t = \alpha$ to p. Clearly, this implies (61) $\square$.

Now we are able to analyze the speed of convergence. The case of single and double roots will be dealt with separately. In the case of single roots, we obtain the following result.

Theorem 1 *If the polynomial p has a root $t^\star$ in $[\alpha, \beta]$ and provided that this root has multiplicity 1, then the sequence of the lengths of the intervals generated by* `quadclip` *which contain that root has the convergence rate $d = 3$.*

Proof 3 The call `quadclip`$(p, [\alpha, \beta])$ generates a sequence of intervals

$$([\alpha_i, \beta_i])_{i=0,1,2,\ldots} \tag{65}$$

with the lengths $h_i = \beta_i - \alpha_i$ whose boundaries converge to $t^\star$. We assume that the first derivative satisfies $p'(t^\star) > 0$. If this assumption is violated, one may consider the polynomial $-p$ instead of p.

Let q_i be the quadratic polynomial obtained by degree reduction with respect to the interval $[\alpha_i, \beta_i]$. Since p' is continuous and due to Lemma 2, the inequalities

$$\|p' - p'(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{4} p'(t^\star) \quad \text{and} \quad \|q_i' - p'\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{4} p'(t^\star) \tag{66}$$

hold for all but finitely many values of i, where the maximum norm refers to the interval $[\alpha_i, \beta_i]$. These two inequalities imply

$$\|q_i' - p'(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{2}\, p'(t^\star), \quad \text{hence} \quad \forall t \in [\alpha_i, \beta_i] :\ q_i'(t) > \frac{1}{2}\, p'(t^\star). \tag{67}$$

On the other hand, the vertical width $2\delta_i$ of the strip enclosed by m and M is bounded by $2\, C_p\, h_i^3$, due to Lemma 1. Thus, the lengths h_i of the intervals

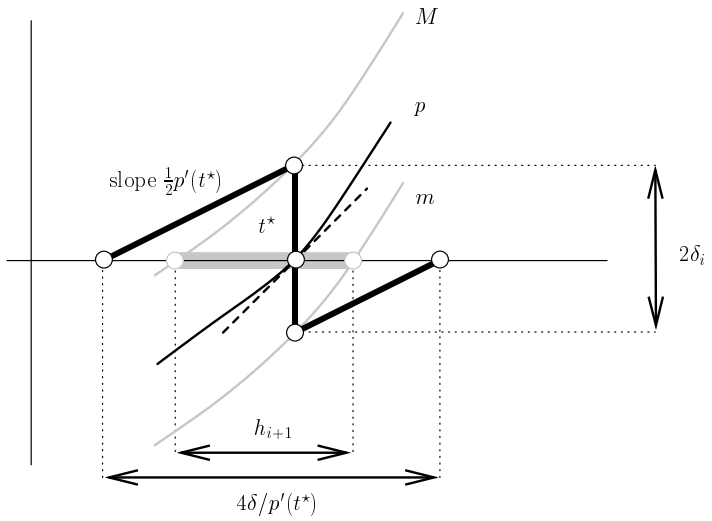


Figure 33: Proof of Eq. (68)

satisfy

$$h_{i+1} \leq \frac{4\,C_p}{p'(t^\star)}\,h_i^3 \tag{68}$$

for all but finitely many values of i, see Fig. 33. □

As for Bézier clipping, multiple roots slow down the speed of convergence. However, the rate is still super-linear for double roots, as described in the following Theorem. See Figure 34 for an illustration.

Theorem 2 *If the polynomial p has a root $t^\star$ in $[\alpha, \beta]$ and provided that this root has multiplicity 2, then the sequence of the lengths of the intervals generated by* `quadclip` *which contain that root has the convergence rate $d = \frac{3}{2}$.*

Proof 4 Similar to the proof of the previous Theorem, we analyze the sequence (65) of intervals with lengths h_i generated by the algorithm which

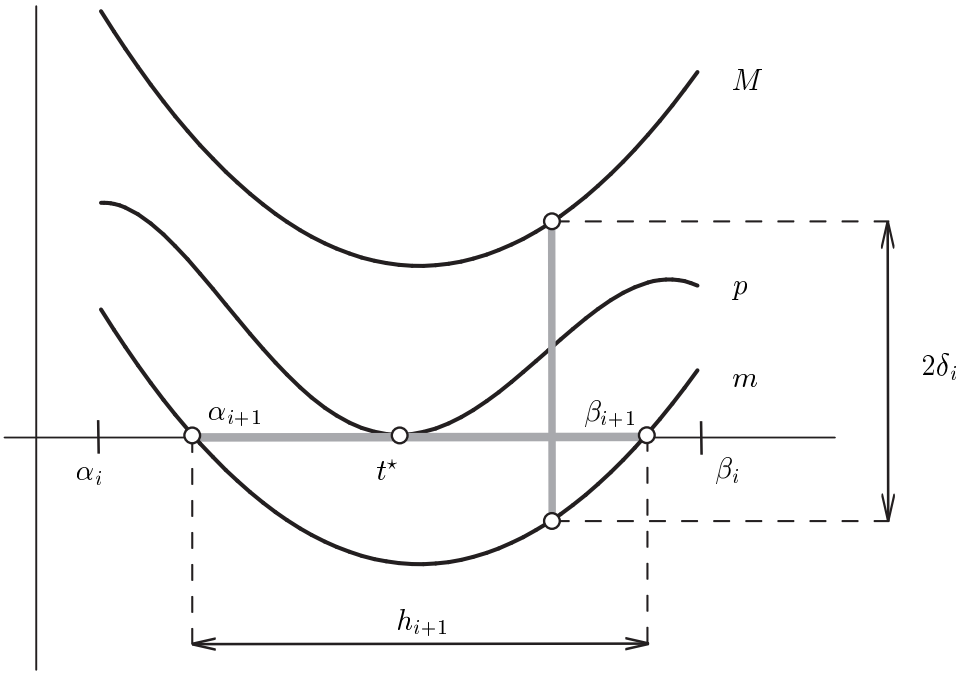


Figure 34: The convergence rate in the double root: The roots of lower bound m define the new interval $[\alpha_{i+1}, \beta_{i+1}]$.

contain the double root. We assume that the second derivative satisfies $p''(t^\star) > 0$. If this assumption is violated, one may again consider the polynomial $-p$ instead of p.

Again, let q_i be the quadratic polynomial obtained by degree reduction with respect to the interval $[\alpha_i, \beta_i]$, and let δ_i be the associated distance bound obtained in line 3 of the algorithm. Since p'' is continuous and due to Lemma 2, the inequalities

$$\|p'' - p''(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \leq \frac{1}{4}p''(t^\star) \quad \text{and} \quad \|q_i'' - p''\|_\infty^{[\alpha_i,\beta_i]} \leq \frac{1}{4}p''(t^\star) \tag{69}$$

hold for all but finitely many values of i, where the maximum norm refers to

the interval $[\alpha_i, \beta_i]$. These two inequalities imply

$$\|q_i'' - p''(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{2}p''(t^\star), \quad \text{hence} \quad \forall t \in [\alpha_i, \beta_i] : q_i''(t) > \frac{1}{2}p''(t^\star). \quad (70)$$

We consider the lower bound $m_i = q_i - \delta_i$ obtained by applying degree reduction with respect to the interval $[\alpha_i, \beta_i]$. Due to $p''(t^\star) > 0$, its intersections with the t–axis bound the next interval $[\alpha_{i+1}, \beta_{i+1}]$ for all but finitely many values of i. Let

$$m_i = \frac{a_i}{2}(t - t^\star)^2 + b_i(t - t^\star) + c_i \quad (71)$$

with certain real coefficients $a_i = q_i''(t^\star)$, $b_i = q_i'(t^\star)$ and c_i. According to (70), the leading coefficient satisfies

$$a_i \ge \frac{1}{2}p''(t^\star) \quad (72)$$

for all but finitely many values of i. Due to the two Lemmas and to $p'(t^\star) = 0$, the other two coefficients satisfy

$$|b_i| = |p'(t^\star) - q'(t^\star)| \le \|p' - q'\|_\infty^{[\alpha_i,\beta_i]} \le D_p\, h_i^2 \quad (73)$$

and

$$\begin{aligned} |c_i| = |p(t^\star) - m(t^\star)| &\le |p(t^\star) - q(t^\star)| + |q(t^\star) - m(t^\star)| \\ &\le \|p - q\|_\infty^{[\alpha_i,\beta_i]} + \delta_i \le (V_p + C_p)\, h_i^3. \end{aligned} \quad (74)$$

The coefficients c_i are non–positive, $c_i \le 0$.

For all but finitely many values of i, the lengths of the interval $[\alpha_{i+1}, \beta_{i+1}]$ are bounded by the difference of the roots of the lower bound m_i, which leads to

$$h_{i+1} \le 2\sqrt{\frac{b_i^2}{a_i^2} - \frac{2c_i}{a_i}} \le \frac{|b_i|}{a_i} + \sqrt{\frac{2|c_i|}{a_i}} \le \frac{2D_p}{p''(t^\star)}h_i^2 + \sqrt{\frac{4(C_p + V_p)}{p''(t^\star)}}\, h_i^{3/2}. \quad (75)$$

Hence, the sequence $(h_i)_{i=0,1,2,\ldots}$ has the convergence rate $\frac{3}{2}$. $\square$

4 Comparison

We compare the two algorithms (Bézier clipping and quadratic clipping) with respect to five criteria: convergence rate, number of operations per iteration step, time per iteration step, number of iterations needed to achieve a certain prescribed accuracy, and computing time needed to achieve a certain prescribed accuracy.

4.1 Convergence rates, number of operations and time per iteration step

The results concerning the *convergence rates* are summarized in Table 2.

With respect to these rates, the new algorithm clearly performs better than Bézier clipping. However, the computational effort per iteration step is equally important. For instance, it is known that solving univariate equations by the secant method, where the convergence rate is $(1+\sqrt{5})/2 \approx 1.618$ for a single root, is generally faster than Newton's method with quadratic convergence rate, since it needs only one evaluation of the function per iteration step, while Newton's method needs one evaluation of the function and and another one of the derivative. Consequently, the computational costs of two steps of the secant method and of one step of the Newton method are comparable.

Table 3 shows the *number of operations* needed per iteration step, where it is assumed that one new interval is found (i.e., $k = 1$ in line 9 of Algorithm 3) and that this interval has shrunk by more than $\frac{1}{2}$, cf. line 5 of Algorithm 1 and line 10 of Algorithm 3. Also, the number of operations needed for computing the convex hull for Algorithm 1 varies slightly; here we assume to

root multiplicity	single root	double root	triple root, etc.
`quadclip`	3	$\frac{3}{2}$	1
`bezclip`	2	1	1

Table 2: Convergence rates of the algorithms `quadclip` and `bezclip`.

degree	`quadclip`						`bezclip`					
n	$\pm$	$*\div$	$\leq$	$\sqrt{}$	$\lvert.\rvert$	$\sum$	$\pm$	$*\div$	$\leq$	$\sqrt{}$	$\lvert.\rvert$	$\sum$
2	120	75	30	4	0	229	90	30	5	0	0	125
4	228	115	32	4	6	385	214	62	9	0	0	285
8	548	243	30	4	2	827	582	174	17	0	0	773
16	1676	691	30	4	2	2403	1698	590	33	0	0	2321

Table 3: Number of operations per step of the iteration for various values of the degree n.

have a convex control polygon, since this is the limit case in general.

The classical Bézier clipping has a slight advantage, though the computational costs of both methods are roughly comparable. The number of operations grows *quadratically* with the degree n. For both algorithms, the computational effort grows linearly, except for the quadratic grow caused by de Casteljau's algorithm which is used to generate the Bernstein–Bézier representation with respect to the newly generated interval. For large degrees n, the de Casteljau algorithm dominates the overall computational costs and the computational costs of both algorithms become increasingly similar.

This picture becomes even more clear by comparing the computation times. We implemented both algorithms in C on a PC with a Intel(R) Xeon(TM)

degree of the polynomial	2	4	8	16
`quadclip`	2.0	2.8	4.4	9.6
`bezclip`	1.3	1.9	3.5	8.3

Table 4: Time per iterations in microseconds for various degrees n.

CPU (2.40GHz) with 512KB of RAM running Linux and measured the time needed for 10^5 iterations (in order to obtain a measurable quantity). The results are reported in Table 4.

In addition, Fig. 35a shows the relation between computing times and polynomial degree, and Fig. 35b visualizes the ratio $t_{\texttt{quadclip}}/t_{\texttt{bezclip}}$. For large values of the degree n, the ratio tends to 1, since the computational effort of the de Casteljau algorithm becomes increasingly dominant.

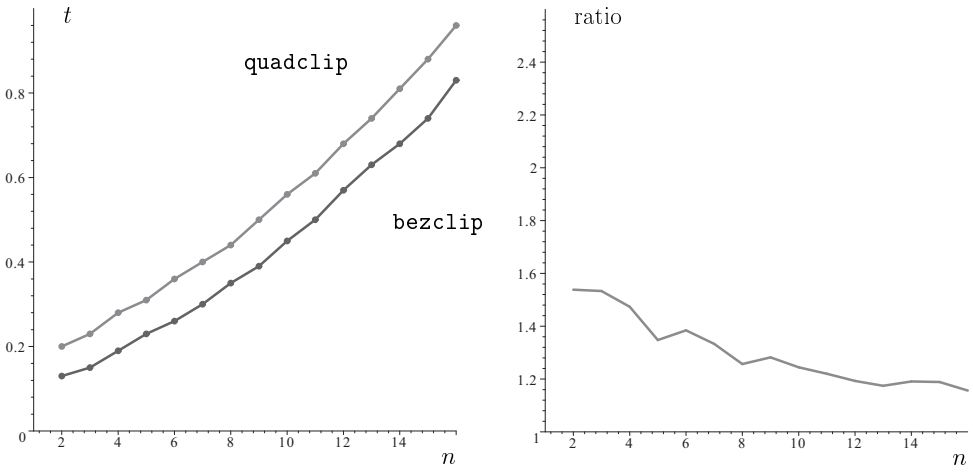
4.2 Number of iterations and computing times vs. accuracy

In order to analyze the relation between the computational effort and the desired accuracy, we discuss three examples, which represent polynomials with a single root, a double root, and two roots which are very close ("near double root").

Example 6 (Single root) We applied the algorithms `bezclip` and `quadclip` to the four polynomials

$$f_2(t) = (t - \tfrac{1}{3})(3 - t), \quad f_4(t) = (t - \tfrac{1}{3})(2 - t)(t + 5)^2,$$
$$f_8(t) = (t - \tfrac{1}{3})(2 - t)^3(t + 5)^4, \quad f_{16}(t) = (t - \tfrac{1}{3})(2 - t)^5(t + 5)^{10}$$

in order to compute the single root $\frac{1}{3}$ in the interval $[0, 1]$. Table 5 reports the number of iterations and the computing times for various values of the



(a) (b)

Figure 35: a) Time per 10^5 iterations of algorithms `quadclip` and `bezclip`. b) Ratio of computing times of both algorithms.

desired accuracy ε. The numbers of iterations were obtained from an implementation in Maple, while the computing times were measured with the help of the implementation in C. The computing times for accuracy below 10^{-16} were obtained by multiplying the number of iterations with the time per iteration (see Table 4). In addition, Figure 37 visualizes the relation between computing times and desired accuracy.

For these four polynomials, the new algorithm (`quadclip`) performs slightly better than Bézier clipping, though the difference is not that significant: the overall computing times to achieve a certain accuracy are roughly the same. In particular, this is true for the realistic range of accuracy (no more than 16 significant digits). This is due to the fact that the quadratic convergence rate of Bézier clipping is already very fast.

Example 7 (Double root) We applied the algorithms `bezclip` and `quadclip`

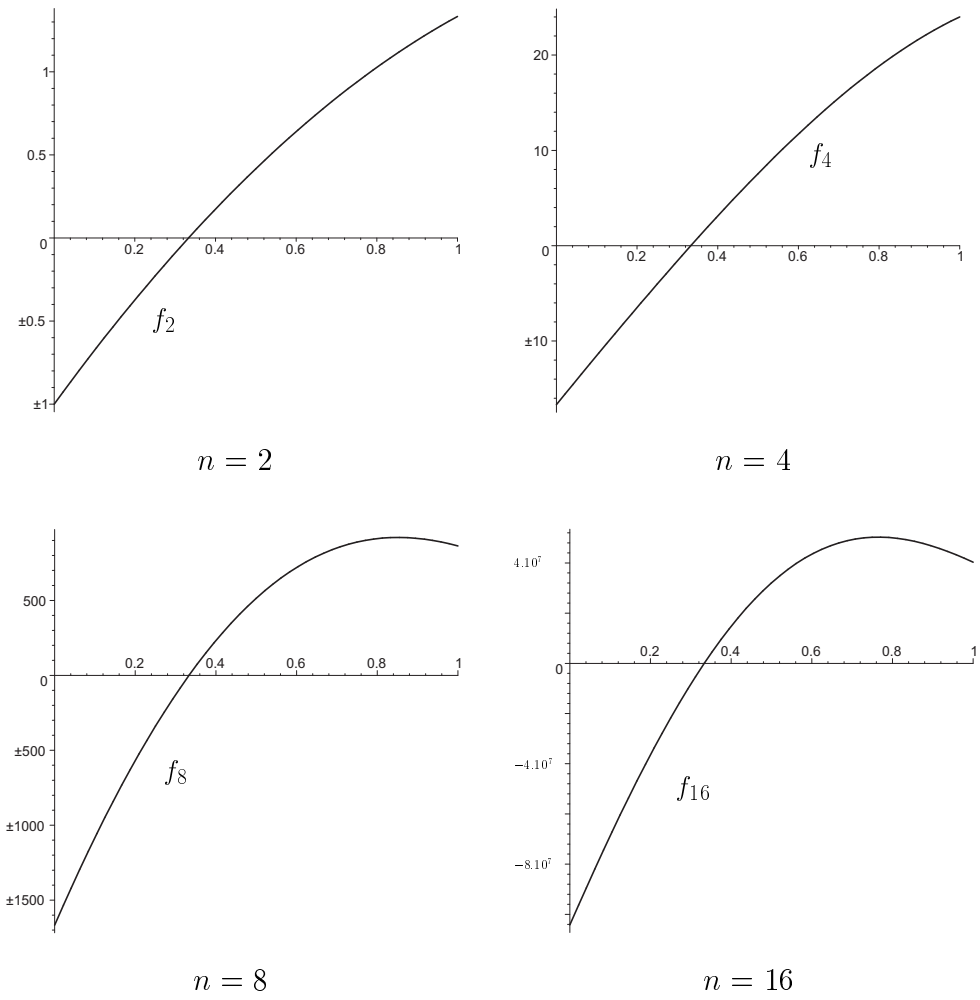


Figure 36: Example 6 (single root): Tested polynomials of degrees 2, 4, 8 and 16 with the root $\frac{1}{3}$ in [0,1].

degree n	ε	10^{-2}		10^{-4}		10^{-8}		10^{-16}		10^{-32}		10^{-64}		10^{-128}	
		quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip
2	N	1	2	1	3	1	3	1	4	1	5	1	6	1	7
	t	2.0	2.5	2.0	3.5	2.0	3.5	2.0	5.9	*2.0*	*7.2*	*2.0*	*8.6*	*2.0*	*9.9*
4	N	2	2	2	3	3	4	3	5	4	6	5	7	5	8
	t	5.4	3.9	5.4	5.5	8.1	7.2	8.2	8.8	*10.8*	*10.6*	*13.4*	*12.5*	*13.5*	*14.4*
8	N	2	2	2	3	3	4	3	5	4	6	5	7	5	8
	t	8.7	6.8	8.9	10.1	13.0	16.9	13.0	20.4	*17.5*	*23.8*	*21.8*	*23.8*	*21.8*	*27.4*
16	N	2	2	2	3	3	4	3	5	4	6	5	7	5	8
	t	18.7	16.3	18.7	24.2	28.0	32.3	28.1	39.9	*37.5*	*47.5*	*46.9*	*55.4*	*46.9*	*63.3*

Table 5: Example 6 (single root): Number of iterations N and computing time t in μs for various values of degree n and accuracy ε. The times for more than 16 significant digits (shown in italic) have been obtained by extrapolation.

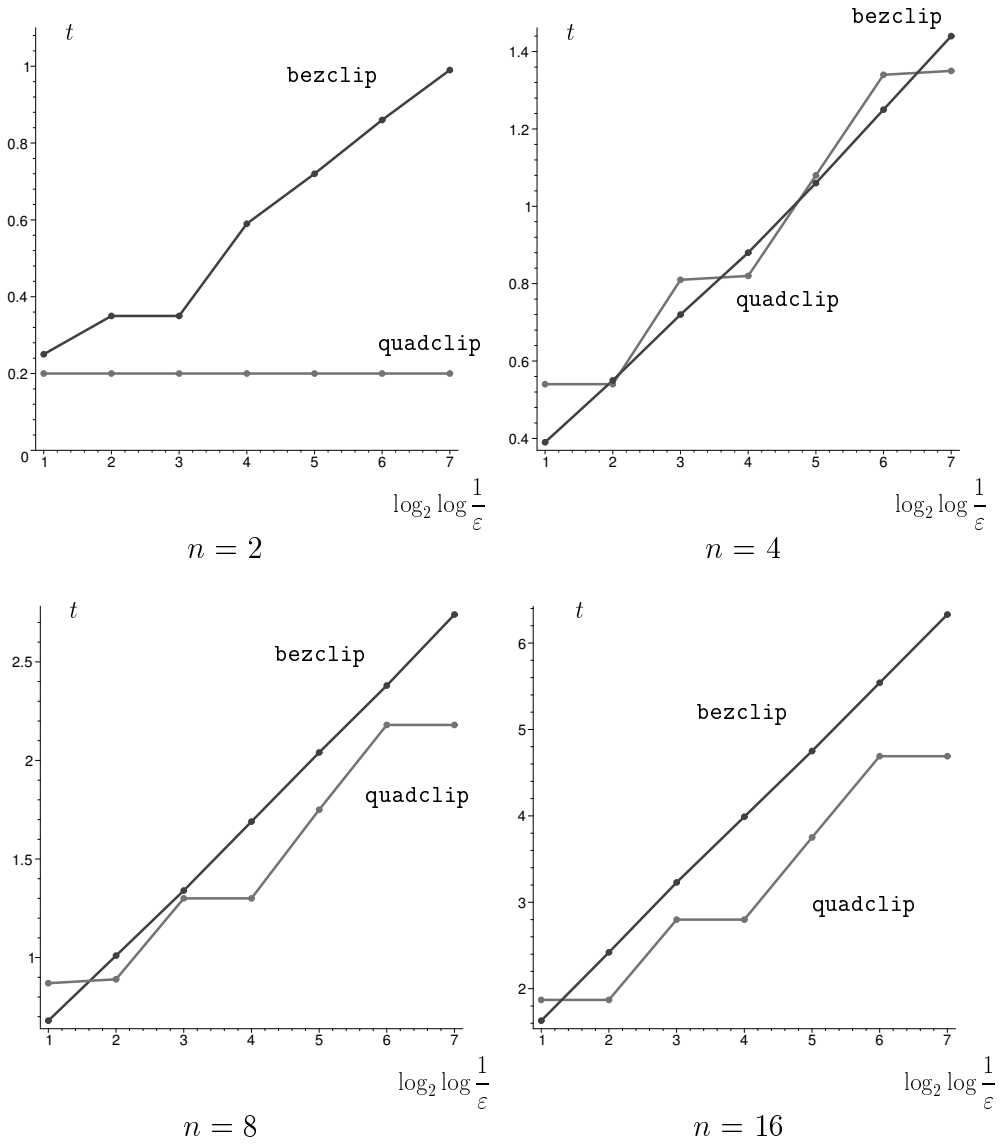
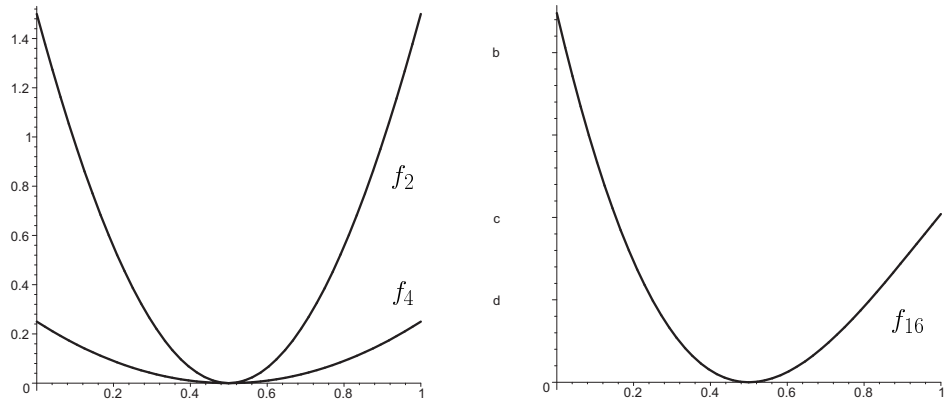


Figure 37: Example 6 (single root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation.



$n = 2, 4$ $n = 16$

Figure 38: Example 7 (double root): Tested polynomials f_2, f_4 and f_{16} with the root $\frac{1}{2}$ on [0,1].

to the four polynomials

$$f_2(t) = (t - \tfrac{1}{2})^2, \quad f_4(t) = (t - \tfrac{1}{2})^2(t + 2)(3 - t),$$
$$f_8(t) = (t - \tfrac{1}{2})^2(4 - t)^3(t + 5)^2(t + 7),$$
$$f_{16}(t) = (t - \tfrac{1}{2})^2(4 - t)^7(t + 5)^6(t + 7)$$

in order to compute the double root $\frac{1}{2}$ in the interval $[0, 1]$. Table 6 reports the number of iterations and the computing times for various values of the desired accuracy ε. Again, the numbers of iterations were obtained from an implementation in Maple, while the computing times were measured with the help of the implementation in C. The computing times for accuracy below 10^{-16} were obtained by multiplying the number of iterations with the time per iteration (see Table 4). In addition, Figure 39 visualizes the relation between computing times and desired accuracy.

For these four polynomials, the new algorithm (`quadclip`) performs far better

degree n	ε	10^{-2}		10^{-4}		10^{-8}		10^{-16}		10^{-32}		10^{-64}		10^{-128}	
		quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip
2	N	1	7	1	14	1	27	1	54	1	107	1	213	1	343
	t	2.0	8.6	2.0	15.6	2.0	30.3	2.0	61.6	*2.0*	*124*	*2.0*	*246*	*2.0*	*383*
4	N	3	7	3	14	4	27	4	53	5	107	7	213	8	332
	t	7.1	13.6	7.2	25.1	10.4	47.2	10.4	93.7	*16.8*	*188*	*19.6*	*375*	*22.4*	*562*
8	N	3	5	4	9	6	17	6	34	9	68	10	135	12	269
	t	12.2	16.7	16.4	32.2	26.6	63.1	26.9	124	*39.6*	*249*	*44.1*	*495*	*52.8*	*988*
16	N	3	4	5	7	6	14	8	27	10	54	11	107	12	213
	t	27.4	32.3	45.4	56.2	56.1	107	76.8	206	*96.2*	*402*	*105*	*823*	*115*	*1635*

Table 6: Example 7 (double root): Number of iterations N and computing time t in μs for various values of degree n and accuracy ε. The times for more than 16 significant digits (shown in italic) have been obtained by extrapolation.

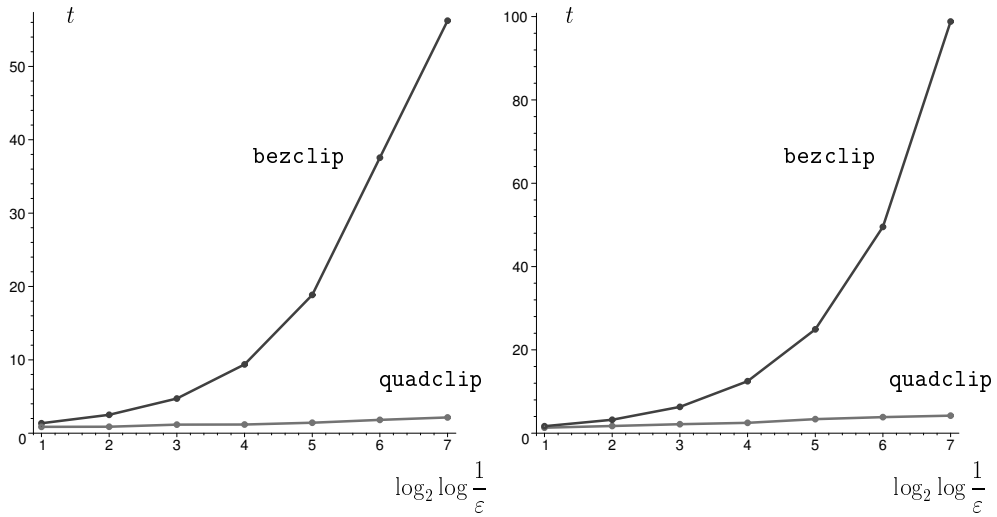


Figure 39: Example 7 (double root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation.

than Bézier clipping. This is due to the higher convergence rate ($\frac{3}{2}$) of the new algorithm.

In practice, the case of doubleroot is unlikely to happen. Conversely, the case of "near double root" is quite frequent and therefore it is worth of scrutinizing.

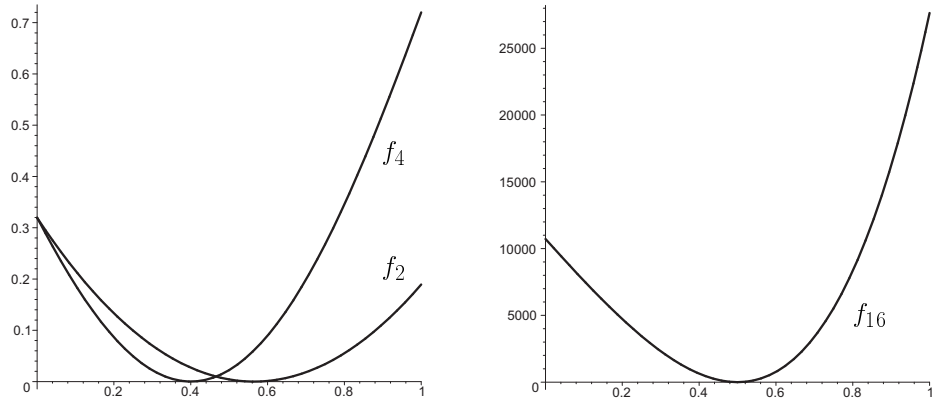


Figure 40: Example 6 (near double root): Tested polynomials f_2, f_4 and f_8 on [0,1].

Example 8 (Near double root) We applied the algorithms `bezclip` and `quadclip` to the four polynomials

$$f_2(t) = (t - 0.56)(t - 0.57), \quad f_4(t) = (t - 0.4)(t - 0.40000001)(t + 1)(2 - t),$$
$$f_8(t) = (t - 0.50000002)(t - 0.50000003)(t + 5)^3(t + 7)^3,$$
$$f_{16}(t) = (t - 0.30000008)(t - 0.30000009)(6 - t)^7(t + 5)^6(t + 7)$$

in order to compute the two roots which are contained within the interval $[0, 1]$. Table 7 reports the number of iterations and the computing times for various values of the desired accuracy ε. Once again, the numbers of iterations were obtained from an implementation in Maple, while the computing times were measured with the help of the implementation in C. The computing times for accuracy below 10^{-16} were obtained by multiplying the number of iterations with the time per iteration (see Table 4). In addition, Figure 39 visualizes the relation between computing times and desired accuracy.

degree n	ε	10^{-2}		10^{-4}		10^{-8}		10^{-16}		10^{-32}		10^{-64}		10^{-128}	
		quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip	quadclip	bezclip
2	N	1	13	1	18	1	20	1	22	1	25	1	27	1	29
	t	2.0	13.2	2.0	18.6	2.0	20.9	2.0	23.1	*2.0*	*24.6*	*2.0*	*247.0*	*2.0*	*29.0*
4	N	3	7	4	13	6	27	8	35	10	37	12	39	14	43
	t	7.1	14.2	9.4	26.9	15.1	52.2	23.9	68.4	*28.1*	*71.8*	*33.6*	*75.3*	*39.2*	*83.6*
8	N	4	5	5	9	7	18	9	26	11	28	13	30	15	32
	t	16.2	20.2	20.3	35.8	30.4	71.4	40.2	103	*49.4*	*111*	*57.4*	*119*	*66.2*	*127*
16	N	2	4	3	7	5	14	7	22	9	24	11	26	11	28
	t	18.6	32.2	27.4	58.4	50.6	113	63.2	176	*86.4*	*192*	*105*	*208*	*105*	*224*

Table 7: Example 8 (near double root): Number of iterations N and computing time t in μs for various values of degree n and accuracy ε. The times for more than 16 significant digits (shown in italic) have been obtained by extrapolation.

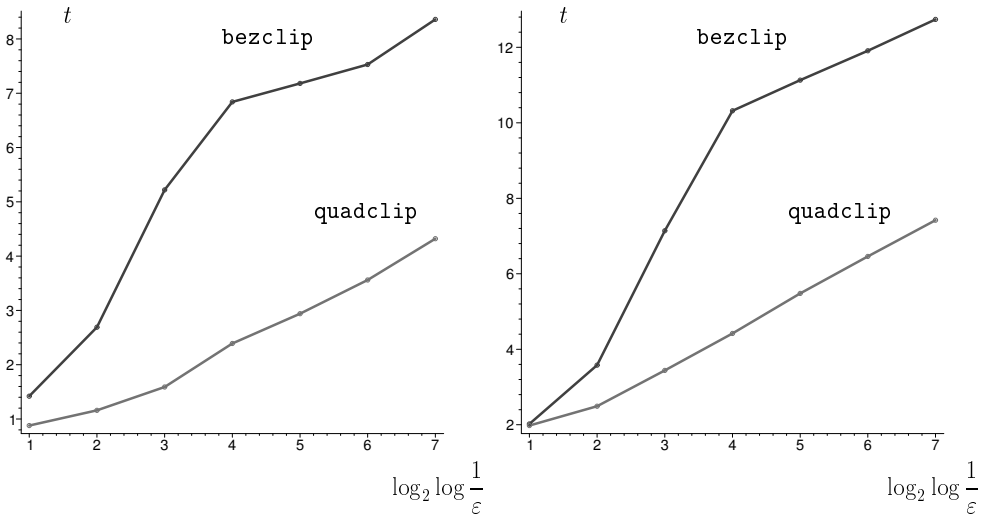


Figure 41: Example 8 (near double root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation.

For these four polynomials, the new algorithm (`quadclip`) performs better than Bézier clipping, since `bezclip` achieves quadratic convergence only after the roots have been separated. Similar effects can be observed if the graph of the polynomial gets very close to the t axis without intersecting it (two or more close conjugate–complex roots).

5 Bivariate linear clipping

In this section, we present the generalization of univariate polynomial solver `quadclip`. Based on degree reduction to *linear* approximants, the new linear clipping algorithm (`bilinclip`) is presented on bivariate polynomial system, however the idea may be easily applied to the system of n equations.

5.1 The root–finding problem

Let us assume system of two polynomial equations

$$\begin{aligned} p(x,y) &= 0 \\ q(x,y) &= 0 \end{aligned} \tag{76}$$

in variables x, y. Let both p and q be polynomials of bidegree (m,n)

$$p(x,y) = c_{m,n}x^m y^n + c_{m-1,n}x^{m-1}y^n + \cdots + c_{0,0}, \quad c_{m,n} \neq 0. \tag{77}$$

Let us denote $\Pi^{m,n}$ the $(n+1)(m+1)$ dimensional linear space of polynomials of bidegree at most (m,n), with the basis $B^{m,n}_{i,j} = \{B^m_i(x)B^n_j(y)\}_{i=0,\ldots,m,j=0,\ldots,n}$, where

$$B^m_i(x) = \binom{m}{i} \frac{(x-\alpha)^i(\beta-x)^{m-i}}{(\beta-\alpha)^m}, \tag{78}$$

$$B^n_j(y) = \binom{n}{j} \frac{(y-\gamma)^j(\delta-y)^{n-j}}{(\delta-\gamma)^n} \tag{79}$$

are the Bernstein polynomials with respect to intervals $[\alpha,\beta],[\gamma,\delta] \subset \mathbb{R}$, respectively. Any polynomial $p \in \Pi^{m,n}$ can be described by its Bernstein–

Bézier representation with respect to the domain $[\alpha, \beta] \times [\gamma, \delta]$,

$$p(x, y) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} B_i^m(x) B_j^n(y), \quad [x, y] \in [\alpha, \beta] \times [\gamma, \delta], \tag{80}$$

with certain Bernstein–Bézier (BB) coefficients $b_{ij} \in \mathbb{R}$.

We consider a given polynomial system (76) in Bernstein–Bézier representation with respect to the domain $[\alpha, \beta] \times [\gamma, \delta]$. All roots of (76) within this domain are to be found. More precisely, we want to generate a set of domains of maximum diameter 2ε which contain the roots, where the parameter ε specifies the desired accuracy.

5.2 The generalization of L_2 norm and dual basis

Following the section 2.4, the definition of L_2 norm and dual basis is easily adopted for bivariate polynomials.

5.2.1 L_2 norm

We consider the space $\Pi^{m,n}$ with the L^2 inner product

$$\langle f, g \rangle^D = \int_{\alpha}^{\beta} \int_{\gamma}^{\delta} f(x, y)\, g(x, y)\, \mathrm{d}y\, \mathrm{d}x \tag{81}$$

with respect to the domain $D = [\alpha, \beta] \times [\gamma, \delta]$ and the norm

$$\|f\|_2^D = \frac{1}{k} \sqrt{\langle f, f \rangle^D}, \tag{82}$$

where $h = (\beta - \alpha)(\delta - \gamma)$, induced by it.

Similarly to the univariate case, the factor $1/h$ is introduced in order to obtain a norm which is *invariant under affine transformations* in the directions of the x and y–axes. More precisely, for any affine transformation

$$\mathcal{A} : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} + \begin{pmatrix} b_{00} & 0 \\ 0 & b_{11} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{83}$$

with $\det(B) \neq 0$ the norms of f with respect to the domain D and of $f \circ \mathcal{A}^{-1}$ with respect to the domain $\mathcal{A}(D)$ are identical,

$$\|f\|_2^D = \|f \circ \mathcal{A}^{-1}\|_2^{\mathcal{A}(D)}. \tag{84}$$

5.2.2 Dual basis

Applying degree reduction with respect to L_2 norm to the given polynomial p gives the unique polynomial $\overline{p} \in \Pi^k$ which minimizes $\|p - \overline{p}\|_2^D$.

Let us assume $\Pi^{m,n}$ with basis $B_{i,j}^{m,n}$ and let Π^k be the $2(k+1)$-dimensional subspace of all polynomials p, of which power of both variables x and y is at most k. Let us assume "monomial" basis $\{\mathbb{B}_i\}_{i=0}^{2k+1} = \{1, x, y, xy, \ldots, x^k y^k\}$.

The *dual basis* $\{\mathbb{D}_i\}_{i=0}^{2k+1}$ to the basis $\{\mathbb{B}_i\}_{i=0}^{2k+1}$ of Π^k is uniquely defined by formula

$$\langle \mathbb{B}_i, \mathbb{D}_j \rangle^D = \delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}, \quad i, j = 0, \ldots, 2k+1. \tag{85}$$

The polynomials $\mathbb{D}_j$ can be represented with respect to the original basis

$$\mathbb{D}_i(x, y) = \frac{1}{h} \sum_{j=0}^{2k+1} r_{i,j} \, \mathbb{B}_j(x, y), \quad i = 0, \ldots, 2k+1. \tag{86}$$

According to the dual basis properties (85), coefficients $r_{i,j}$ are easy to compute via linear system of equations.

The polynomial $\overline{p}$ obtained by applying degree reduction to p (see (80)) with respect to the domain D may be computed from

$$\overline{p}(x,y) = \sum_{j=0}^{2k+1} \langle p, \mathbb{D}_j \rangle^D \, \mathbb{B}_j = \sum_{j=0}^{2k+1} \left(\sum_{i=0}^{m} \sum_{l=0}^{n} b_{i,l} \beta_{i,l}^j \right) \mathbb{B}_j \,, \tag{87}$$

with the coefficients

$$\beta_{i,l}^j = \langle B_{i,l}^{m,n}, \mathbb{D}_j \rangle^D. \tag{88}$$

Example 9 Computing coefficients $r_{i,j}$, the equation (86) is multiplied (with respect to the L^2 inner product (82)) by $\mathbb{B}_i$, $i = 0, 1, 2$. Solving the linear system, the dual basis $\{\mathbb{D}_j\}_{j=0}^2$ to the basis $\{\mathbb{B}_i\}_{i=0}^2 = \{1, x, y\}$ on the unit domain $\langle 0,1\rangle \times \langle 0,1\rangle$ is:

$$\begin{aligned} \mathbb{D}_0 &= 7 - 6x - 6y \\ \mathbb{D}_1 &= -6 + 12x \\ \mathbb{D}_2 &= -6 + 12y. \end{aligned} \tag{89}$$

Example 10 The matrix of reduction coefficients for $m = 3$, $n = 3$.

$$(\beta_{i,j}^0) = \begin{bmatrix} \frac{23}{28} & \frac{17}{28} & \frac{11}{28} & \frac{1}{16} \\ \frac{17}{80} & \frac{11}{80} & \frac{1}{16} & -\frac{1}{80} \\ \frac{11}{80} & \frac{11}{16} & -\frac{1}{80} & -\frac{7}{80} \\ \frac{1}{16} & -\frac{1}{80} & -\frac{7}{80} & -\frac{13}{80} \end{bmatrix}$$

The best linear approximant $\overline{p}$ to p with respect to the L^2 norm is

$$\overline{p}(x,y) = \delta_0 + \delta_1 x + \delta_2 y, \tag{91}$$

where

$$\delta_k = \sum_{i=0}^{3} \sum_{j=0}^{3} b_{i,j} \beta_{i,j}^k, \quad k = 0, 1, 2 \tag{92}$$

and $b_{i,j}$ are the Bernstein-Bézier coefficients of p (see (80)).

5.3 Algorithm

Some steps of the algorithm will be explained in more detail:

- In line 2 of the algorithm, we generate the best linear approximant $\overline{p}$ of p with respect to the L^2 norm on the current domain $[\alpha, \beta] \times [\gamma, \delta]$. In other words, coefficients δ_0, δ_1, δ_2 are to be found such that

$$I = \frac{1}{h} \int_{\alpha}^{\beta} \int_{\gamma}^{\delta} p(x, y) - (\delta_0 + \delta_1 x + \delta_2 y) \, \mathrm{d}y \, \mathrm{d}x \tag{93}$$

  is minimal. This is achieved via dual basis (see Example 10 and Fig. 42(d)). The linear approximant $\overline{q}$ is found the same way.

- In order to obtain the bound δ^p on

$$\|p - \overline{p}\|_{\infty}^{D} = \max_{[x,y] \in D} |p(x, y) - \overline{p}(x, y)|, \tag{94}$$

  see line 3, we represent the linear function $\overline{p}$ as a Bernstein–Bézier patch of bidegree (m, n). Therefore, $\overline{p}$ is represented via $(m+1)(n+1)$ control points $c_{i,j}$, $i = 0, \ldots, m, j = 0, \ldots, n$. Due to the properties of Bernstein polynomials, the bound is computed as

$$\delta^p = \max_{i,j} |b_{i,j} - c_{i,j}|, \quad i = 0, \ldots, m, j = 0, \ldots, n \tag{95}$$

  see Fig. 43(e), where $b_{i,j}$ and $c_{i,j}$ are the coefficients of the Bernstein–Bézier representations of p and $\overline{p}$ with respect to domain D, respectively.

Algorithm 3 `bilinclip`$(p,\ q,\ [\alpha,\beta],\ [\gamma,\delta])$ {Bivariate linear clipping}

if diameter of domain $[\alpha,\beta]\times[\gamma,\delta] = D \geq \varepsilon$ **then**
  $\overline{p}$, $\overline{q}$ $\leftarrow$ generate the best linear approximant to p, q with respect to the L^2 inner product on D.
  δ^p, δ^q $\leftarrow$ compute bound on $\|p-\overline{p}\|_\infty^D$ by comparing the Bernstein–Bézier representations of p and $\overline{p}$.
  $p^U \leftarrow \overline{p} + \delta^p$ {upper bound of p}
  $p^L \leftarrow \overline{p} - \delta^p$ {lower bound of p}
  $q^U \leftarrow \overline{q} + \delta^q$ {upper bound of q}
  $q^L \leftarrow \overline{q} - \delta^q$ {lower bound of q}
  $P \leftarrow$ parallelogram in the plane $z=0$ bounded by lines $p^U=0$, $p^L=0$, $q^U=0$, $q^L=0$.
  $R \leftarrow$ rectangle bounding P.
  if $D\cap R = \emptyset$ **then**
    return $(\emptyset)$
  else
    Define new domain $D' = D\cap R$.
    if $\operatorname{diam}(D') \geq \frac{1}{2}\operatorname{diam}(D)$ **then**
      return (`bilinclip` $(p,\ q,\ [\alpha,\frac{1}{2}(\alpha+\beta)]\times[\gamma,\frac{1}{2}(\gamma+\delta)])$
      $\cup$ `bilinclip` $(p,\ q,\ [\frac{1}{2}(\alpha+\beta),\beta]\times[\gamma,\frac{1}{2}(\gamma+\delta)])$
      $\cup$ `bilinclip` $(p,\ q,\ [\alpha,\frac{1}{2}(\alpha+\beta)]\times[\frac{1}{2}(\gamma+\delta),\delta])$
      $\cup$ `bilinclip` $(p,\ q,\ [\frac{1}{2}(\alpha+\beta),\beta]\times[\frac{1}{2}(\gamma+\delta),\delta])$)
    else
      $S \leftarrow$ `bilinclip`$(p,\ q,\ D')$
      return (S)
    end if
  end if
else
  return (D)
end if

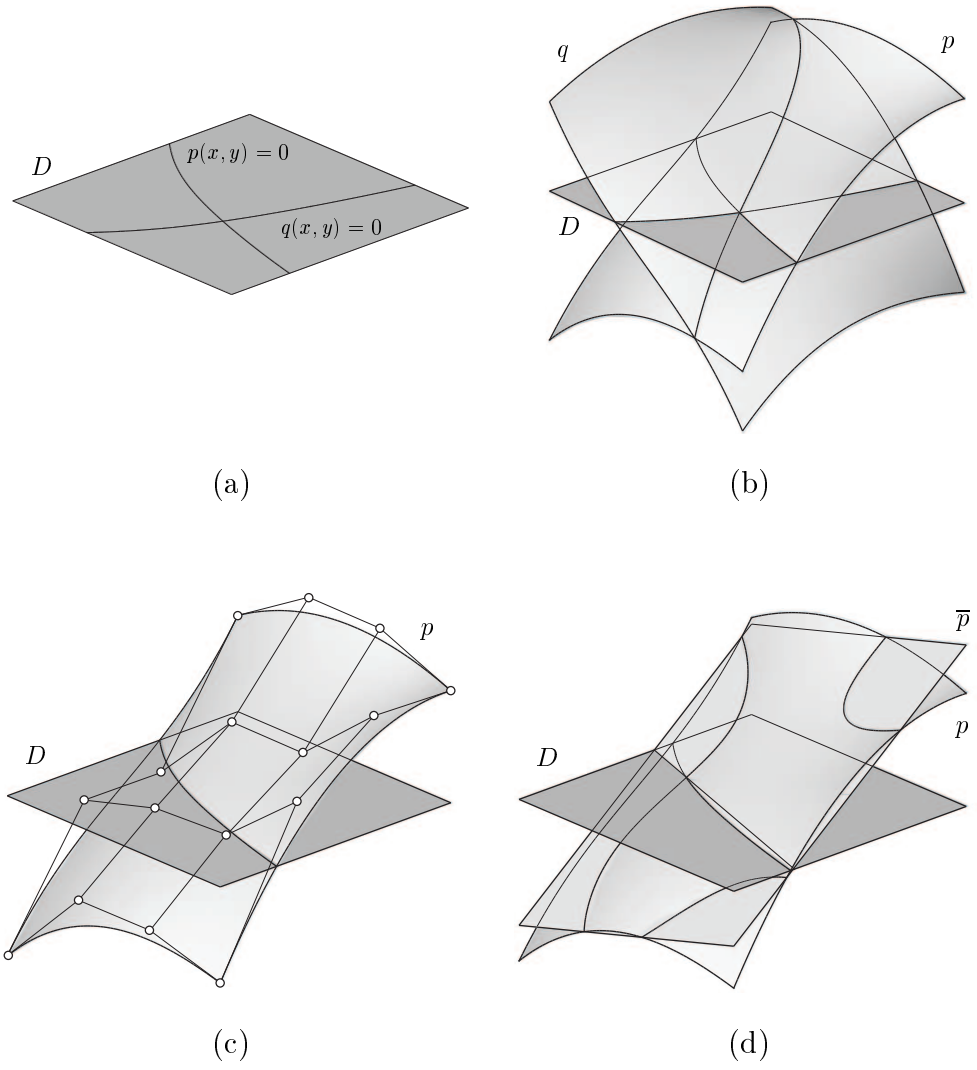


Figure 42: One iteration of `bilinclip` I: (a) the polynomial system (76) over given domain D, (b) the graphs of polynomials $z = p(x, y)$, $z = q(x, y)$ are represented by surfaces in $\mathbb{R}^3$, (c) polynomial p expressed by Bernstein–Bézier patch over D, (d) $\overline{p}$ is the best linear approximant of p with respect to L_2 norm over domain D.

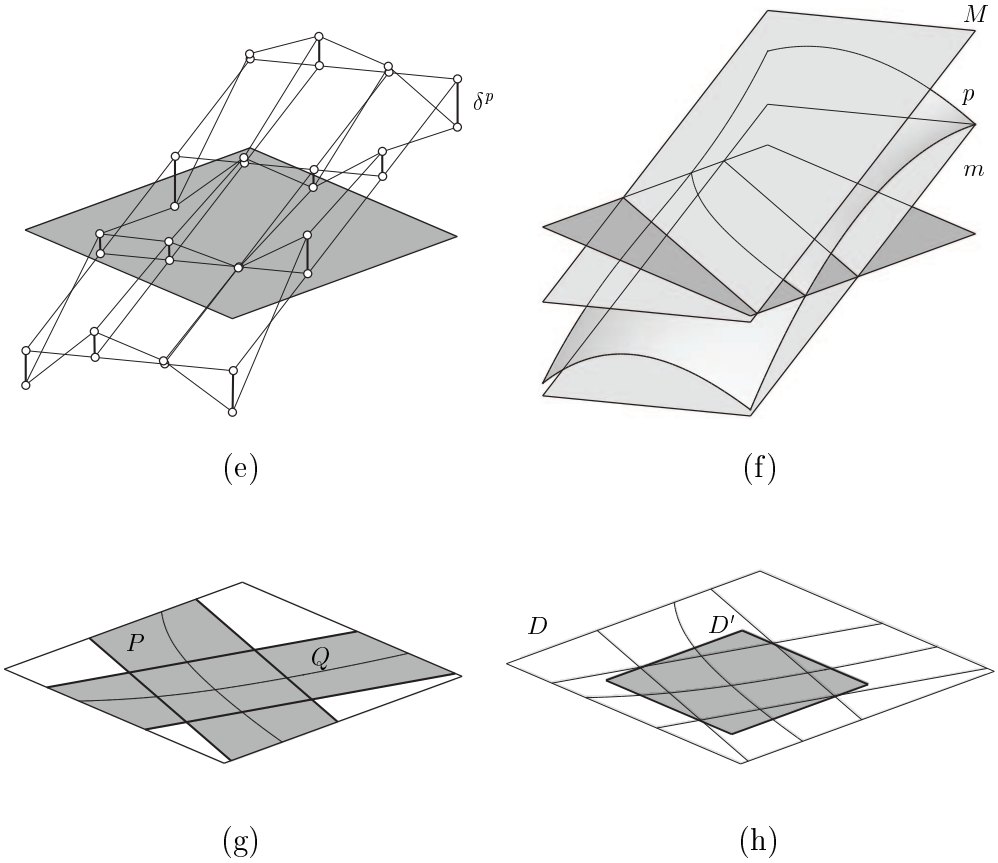


Figure 43: One iteration of `bilinclip` II: (e) p and $\overline{p}$ are both expressed in Bernstein-Bézier form, the error bound δ^p is obtained as the maximum length of the thick black vertical bars, (f) the lower and upper bounds $p^L = \overline{p} - \delta^p$ and $p^U = \overline{p} + \delta^p$, (g) boundaries of polynomials p and q are intersected with plane xy in order to construct planar bounding strips P and Q, respectively, (h) the intersection of strips is bounded by the least square that defines the new domain D'.

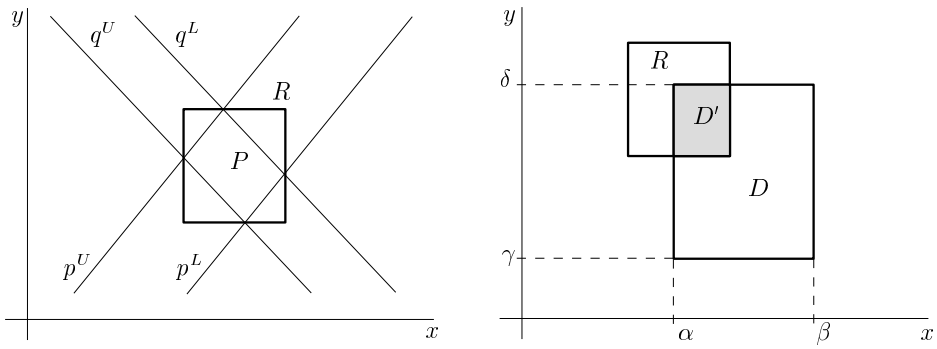


Figure 44: Construction of the new domain: (a) Parallelogram P is bounded by the least rectangle of which sides are parallel with axes x and y, (b) intersection of the original domain D and the rectangle R gives the new domain D'.

- In lines 4 and 5, the bound δ^p is used to construct linear functions $p^U(x, y)$ and $p^L(x, y)$ (Fig. 43(f)) satisfying

$$\forall [x, y] \in D : \quad p^L(x, y) \leq p(x, y) \leq p^U(x, y). \tag{96}$$

- In lines 8 and 9, see Fig. 43(g), the root-containing parallelogram P is constructed and the least rectangle R that includes P is found.

- In lines 10–20 we analyze the mutual position of the original domain D and the rectangle R, see Fig. 44. If the intersection is empty, then no roots exist. Otherwise, we obtain the new domain $D' = D \cap R$.

- If the diameter of this domain D' is sufficiently small, when compared to the length of the diameter of the previous domain D, then `bilinclip` is applied to it (line 17). Otherwise we bisect the original domain D and apply `bilinclip` to the four subdomains (line 15).

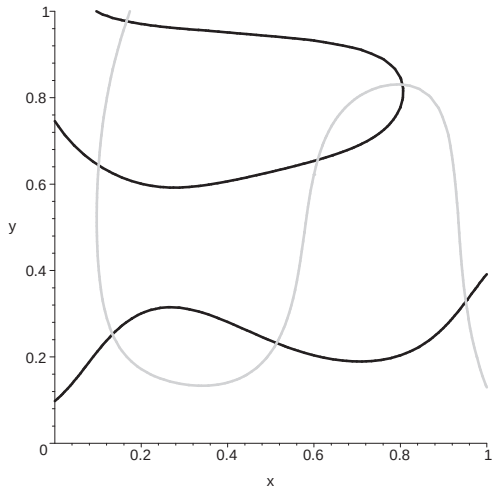


Figure 45: The null set of p and q in unit domain

5.4 Examples

In all examples bellow, all time-values of C code implementation of `bilinclip` were measured on a PC with a Intel(R) Xeon(TM) CPU (2.40GHz) with 1.98GB of RAM running Linux. The loop of 10^4 repetitons was measured (in order to obtain a measurable quantity).

Example 11 We applied the alogorithm `bilinclip` to the polynomials p and q of the bidegree $(5, 5)$ in order to compute all roots within the domain $\langle 0, 1\rangle \times \langle 0, 1\rangle$, see (Fig. 45). $P^{5,5}$ and $Q^{5,5}$ are the matrices of Bernstein–Bézier coefficients of p and q, respectively. The graphs of both polynomial in $\mathbb{R}^3$ are visualized in Fig. 46.

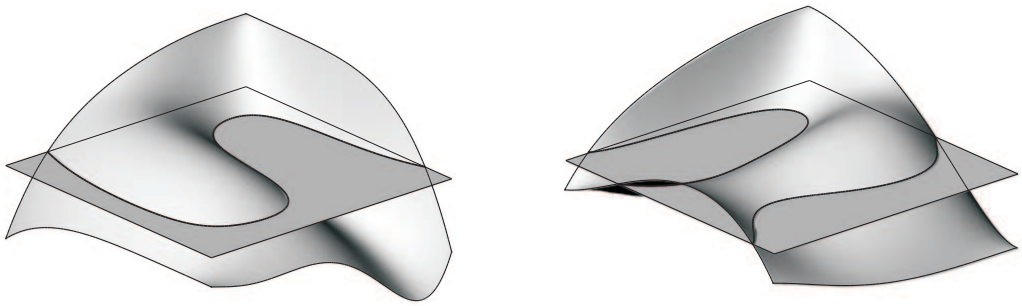


Figure 46: Polynomials p and q over unit domain.

$$P^{5,5} = \begin{bmatrix} -20 & -20 & -20 & -20 & -20 & -50 \\ -30 & 30 & 30 & 30 & 30 & 30 \\ -50 & 50 & 50 & 50 & 50 & 30 \\ 50 & 50 & 50 & 50 & 50 & 50 \\ -150 & -150 & -150 & -150 & -150 & 50 \\ -50 & 50 & 50 & 50 & 50 & 50 \end{bmatrix} \quad Q^{5,5} = \begin{bmatrix} -20 & 30 & 30 & 20 & -10 & -20 \\ -30 & -30 & -10 & 100 & -150 & 30 \\ -50 & -50 & -10 & 100 & -150 & 30 \\ -50 & 50 & 50 & 100 & -150 & 50 \\ -50 & 50 & 50 & 100 & -150 & 50 \\ -50 & -50 & 20 & 30 & 50 & 50 \end{bmatrix}$$

Table 8 shows the numbers of progressive steps (*clipping*), bisection steps, number of all iterations, number of roots and time with respect to the prescribed accuracy ε. According to the shape of both polynomials, bisection steps dominate when low accuracy is required. After separating roots, algorithm works progressively without bisection steps. Achieving sufficiently high accuracy, the phantom root is eliminated.

Example 12 We applied the algorithm `bilinclip` to the Astroid $p = x^{\frac{2}{3}} + y^{\frac{2}{3}} - 5^{\frac{2}{3}}$ and Maltese Cross curve $q = (x^2 + y^2) - xy\,(x^2 - y^2)$ within the

ε	10^{-2}	10^{-4}	10^{-8}	10^{-16}
clipping	19	28	36	50
bisection	12	12	12	12
all iterations	60	70	78	92
roots	8	7	7	7
time(ms)	1.76	2.10	2.39	3.08

Table 8: Example 11; computing time, number of all iterations and number of roots within prescribed accuracy ε.

domain $\langle -5, 5\rangle \times \langle -5, 5\rangle$ (see Fig. 47). Table 9 shows the numbers of progressive steps, bisection steps, number of all iterations, number of roots and time with respect to the prescribed accuracy ε.

Example 13 `bilinclip` was applied to the Descartes leaf $p = x^3 + y^3 - 3\,xy$ and Lemniscat of Bernoulli $q = (x^2 + y^2)^2 - 2\,x^2 + 2\,y^2$ within the domain $\langle -2, 2\rangle \times \langle -2, 2\rangle$, see (Fig. 48). Again, Table 10 gives the numbers of progressive steps, bisection steps, number of all iterations, number of roots and time with respect to the prescribed accuracy ε. With respect to the multipleroot in $[0, 0]$, significantly more iterations are necessary. Realizing the structure of `bilinclip`, domain $\langle -2, 2\rangle \times \langle -2, 2\rangle$ is bisected after first iteration and root $[0, 0]$ is consecutively included in four subdomains. Consequently, six roots is reasonable solution within accuracy 10^{-16}.

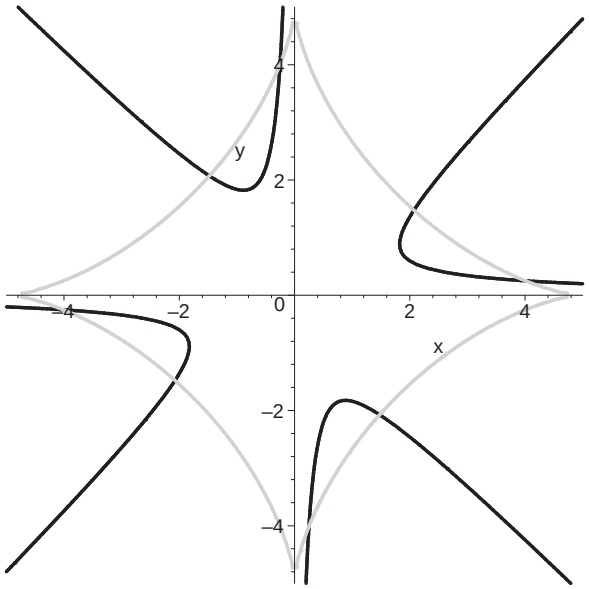


Figure 47: Astroid and Maltese Cross curve on domain $\langle -5, 5\rangle \times \langle -5, 5\rangle$.

ε	10^{-1}	10^{-2}	10^{-4}	10^{-8}	10^{-16}
clipping	32	40	52	60	72
bisection	49	49	49	49	49
all iterations	217	229	241	249	361
roots	12	8	8	8	8
time(ms)	8.43	9.38	9.50	9.81	11.47

Table 9: Example 12; number of iterations with respect to the accuracy ε.

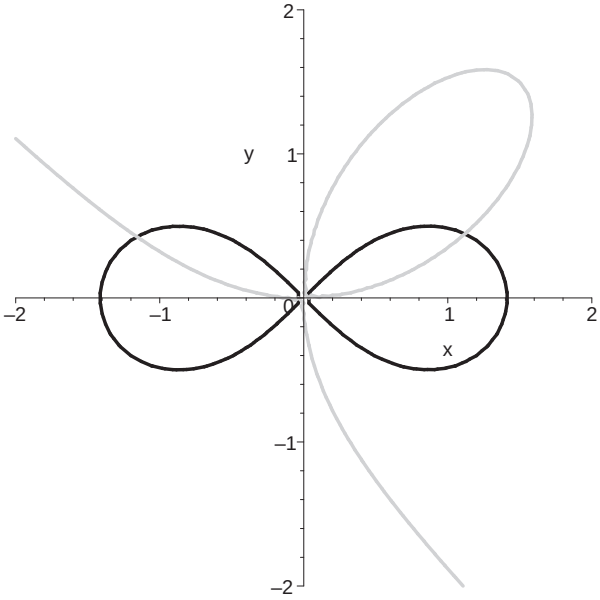


Figure 48: Descartes leaf and Lemniscat of Bernoulli on $\langle -5, 5\rangle \times \langle -5, 5\rangle$

ε	10^{-2}	10^{-4}	10^{-8}	10^{-16}
clipping	12	27	41	47
bisection	50	66	66	66
all iterations	195	282	298	308
roots	18	10	8	6
time(ms)	3.09	4.07	4.39	4.63

Table 10: Example 13; computing time and number of iterations within prescribed accuracy ε

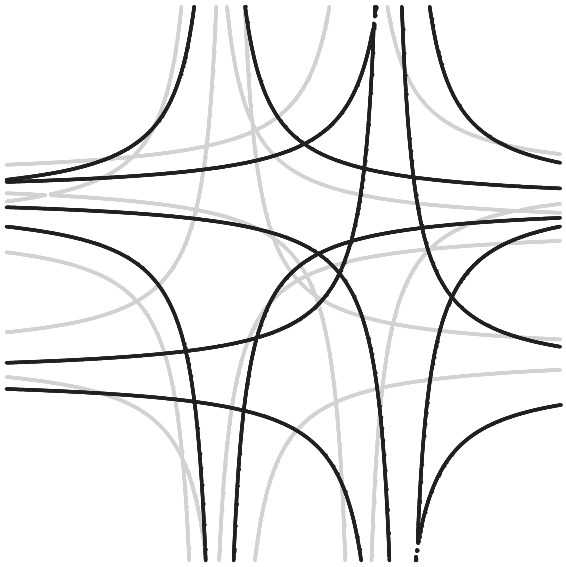


Figure 49: Intersection of curves of bidegree $(6, 6)$

	`bilinclip`	`IPP*`
iter.	2305	389
result	39	78
$time(ms)$	85	44

Table 11: Example 14; time and number of iterations within prescribed accuracy ε

In order to get comparable data with IPP algorithm, we applied our method on two examples presented in Mourrain and Pavone (2005). With respect to computer settings data presented in this paper (Intel Pentium 4, 2.0 GHz with 512 Mo RAM), our time–concerning data should be increased by some "handicap" constant compensating different processor tacting.

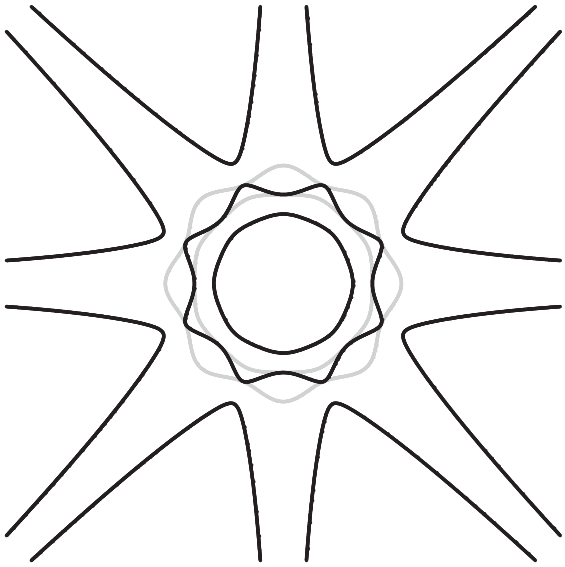


Figure 50: Intersection of curves of bidegree $(8, 8)$

Example 14 Intersection of curves of bidegree $(6, 6)$ are computed, see Fig. 49. Table 11 shows number of iterations, number of roots and time with respect to the accuracy $\varepsilon = 10^{-6}$ computed via `bilinclip` and *interval projected polyhedron algorithm* with local preconditioner (`IPP*`).

Example 15 Intersection of curves of bidegree $(8, 8)$, Mourrain and Pavone (2005) example a, are computed. Table 12 shows number of iterations, number of roots and time with respect to the accuracy $\varepsilon = 10^{-6}$ computed via `bilinclip` and (`IPP*`).

Remark 3 In examples above, the behaviour of `bilinclip` was different with the respect of single or multiple roots. We remind the definition of single root of system (76). We say, that point $A = [x_0, y_0]$ is a single root of system (76) if and only if $(p(x_0, y_0) = 0, q(x_0, y_0) = 0)$ and there exist gradients $\nabla_A q$, $\nabla_A p$ and are linearly independent (see Figure 51).

	`bilinclip`	`IPP*`
iter.	2045	1055
result	16	60
$time(ms)$	76	120

Table 12: Example 15; time and number of iterations within prescribed accuracy ε

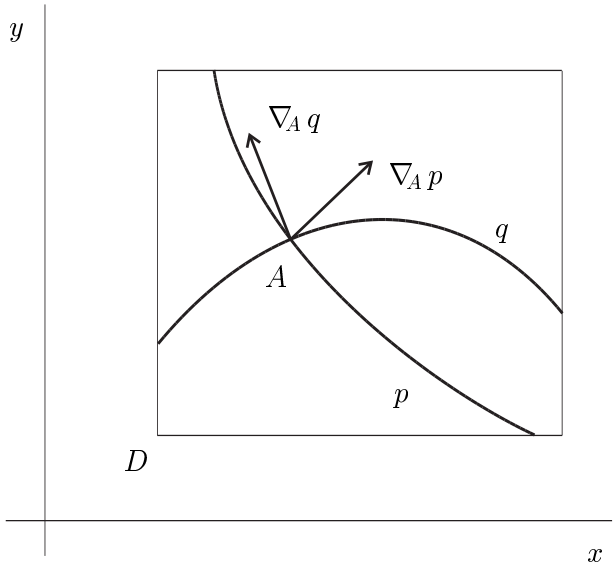


Figure 51: Single root of bivariate polynomial system.

Remark 4 Let us assume the system (76) has only one single root in domain $D = D_0$ and let us denote d_i the diameter of domain D_i after i-th iteration of `bilinclip`. According to many examples, the sequence $(d_i)_{0,1,2,\ldots}$ has the convergence rate $r = 2$. The proof is under research.

Remark 5 In fact, the algorithm `bilinclip` is not pure generalization of `quadclip`. Contrary to `quadclip`, the best approximant with respect to L_2 norm is "only" linear in bivariate case. Using quadratic approximant, one can expect – similarly to univariate case – higher convergence rate. Equally, the number of operations (and consequently the time per one iteration) is expected to be increased. The intersection of two conics instead of lines (compare with line 8, Algorithm 3) appears to be more complicated. Anyway, the improvement of algorithm `bilinclip` could be possible this way and is still under research.

6 Conclusion

Based on the techniques of degree reduction, we derived an algorithm for computing all roots of a given polynomial within a given interval, with a certain accuracy. We analyzed the convergence rates of the new technique and compared it with the classical technique of Bézier clipping. In the case of single roots, the new algorithm performs similarly to Bézier clipping. For double and near double roots, however, it reduces the computational effort. This is due to its superlinear convergence rate $(\frac{3}{2})$ in the case of double roots.

As a direct improvement of the method, one may replace the quadratic polynomial q by a cubic or even a quartic one. In this case, the formulas of Cardano and Ferrari are needed to compute the intersections of the bounding polynomial strip with the t–axis. Clearly, these computations are more involved than in the case of a quadratic polynomial. It is to be expected that such a generalized algorithm would provide an even higher convergence rate for single and double roots, and superlinear convergence for roots with multiplicities 3 and 4.

Demonstrated on bivariate case, the generalization of the algorithm is presented. This technique is based on approximation by linear polynomials and it appears to have quadratic convergence rate in single roots, leading to results comparable to those of Mourrain and Pavone (2005). Attempting to reach faster convergence rate for single roots and superlinear convergence for double roots, we will follow the idea of approximation by polynomials of higher degree.

List of Figures

1 Root-finding problem: all roots of polynomial $f(x)$ on interval $[a, b]$ are required. 2

2 Bivariate polynomial system: polynomial system (1) over domain Ω represented as curves intersection. 3

3 Bivariate polynomial system: solution of the system (1) represented as the intersection of three surfaces: $z = f(x, y)$, $z = g(x, y)$, $z = 0$. 4

4 Bernstein polynomials for $n = 4$. 7

5 Bézier curves of degree 4: curve with b) inflection, c) loop, d) cusp. 8

6 Variation diminishing property: arbitrary line has no more intersections with Bézier curve then with its control polygon. . 10

7 Convex hull property: curve is included inside the convex hull of its control polygon. 11

8 Construction of parabola by linear interpolation. 12

9 de Casteljau algorithm: Construction of a point of Bézier curve for parameter value $t_0 = \frac{3}{5}$. 13

10 Subdivision: Construction of a control polygon of Bézier curve defined on subinterval $[0, \frac{3}{5}]$. 15

11 Degree elevation: Bézier cubic represented by five control points after one elevation step. 16

12 Repeated degree elevation: Parabolic arc after four degree elevation steps. 17

13 Nonparametric curve: polynomial $f(x)$ of degree six on $[0,1]$ represented as a Bézier curve. 18

14 Bilinear Bézier patch: isoparametric curves for $u_0 = \frac{1}{2}$ and $v_0 = \frac{1}{3}$. 21

15 Bézier patch of bidegree $(3,3)$: bounding curves and control point mesh. 22

16 Biquadratic Bézier patch: tangent plane in the corner control point. 23

17 Tensor product surface: shaping u-curve $\mathbf{b}^m(u)$ sweeps along the path v-curve $\mathbf{b}_2^n(v)$. 24

18 Tensor product bicubic patch: construction of a point $\mathbf{b}^{3,3}(\frac{2}{3}, \frac{1}{2})$ via five de Casteljau algorithms. 25

19 Degree elevation of Bézier patch: reduction to several univariate degree elevation steps. 27

20 Nonparametric patch: polynomial $f(x,y)$ represented as Bézier patch. 28

21 The affine transformation: the L^2 norm of f and its affine image $\mathcal{A}(f)$ are the same. 30

22 Maximum norm on BB-coefficients: polynomial f and its norm $\|f\|_{\mathrm{BB},\infty}^{[\alpha,\beta]}$ with respect to the interval $[\alpha,\beta]$. 31

23 Maximum norm: polynomial f and its norm $\|f\|_\infty^{[\alpha,\beta]}$ with respect to the interval $[\alpha, \beta]$. 31

24 Degree reduction: polynomial p and its best approximant q with respect to L^2 norm. 33

25 Dual basis: The Bernstein basis of degree 3 and the associated dual basis functions. 34

26 The best quadratic approximant: given polynomial p and its best approximant q with respect to L^2 norm; y-coordinates of control points $\mathbf{c}_0$, $\mathbf{c}_1$, $\mathbf{c}_2$ are computed via matrix (46); x-coordinates are distributed uniformly on $[\alpha, \beta]$. 37

27 One iteration of `bezclip`: (a) The polynomial p is represented in BB-form on $[\alpha, \beta]$, (b) The convex hull $\mathcal{C}$ of control polygon is constructed, (c) $\mathcal{C}$ is intersected with t–axis in order to define new interval $[\alpha', \beta']$, (d) p is subdivided on $[\alpha', \beta']$. 38

28 False positive answer: If the lenght of the interval $[\alpha', \beta']$ is less then prescribed accuracy ϵ, `bezclip` returns $[\alpha', \beta']$ as root-containing interval. 39

29 One iteration of `quadclip`: (a) the polynomial p is represented in BB-form on $[\alpha, \beta]$, (b) q – the best quadratic approximant of p with respect to L_2 norm, (c) the error bound δ is obtained as the maximum length of the thick grey vertical bars, (d) the lower and upper bounds $m = q - \delta$ and $M = q + \delta$, the intersection of the strip enclosed by them with the t–axis defines the new interval $[\alpha', \beta']$. 43

30 The definition of a new domain: a) The parabolic strip – enclosed by M and m – is intersected with t-axis, five options may occur, b) Case 1, intervals defined by roots of M and m are intersected with the original interval $[\alpha, \beta]$ in order to obtain the new domains. 45

31 The degree-raised parabola: The best quadratic approximant q, its original BB-representation $(\overline{c}_0, \overline{c}_1, \overline{c}_2)$ and the control polygon of q after degree raising. 46

32 The lenghts of both thick grey vertical bars are of $O(h^3)$. . . 47

33 Proof of Eq. (68) 50

34 The convergence rate in the double root: The roots of lower bound m define the new interval $[\alpha_{i+1}, \beta_{i+1}]$. 51

35 a) Time per 10^5 iterations of algorithms `quadclip` and `bezclip`. b) Ratio of computing times of both algorithms. 56

36 Example 6 (single root): Tested polynomials of degrees 2, 4, 8 and 16 with the root $\frac{1}{3}$ in [0,1]. 57

37 Example 6 (single root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation. 59

38 Example 7 (double root): Tested polynomials f_2, f_4 and f_{16} with the root $\frac{1}{2}$ on [0,1]. 60

39 Example 7 (double root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation. 62

40 Example 6 (near double root): Tested polynomials f_2, f_4 and f_8 on [0,1]. 63

41 Example 8 (near double root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation. 65

42 One iteration of `bilinclip` I: (a) the polynomial system (76) over given domain D, (b) the graphs of polynomials $z = p(x, y)$, $z = q(x, y)$ are represented by surfaces in $\mathbb{R}^3$, (c) polynomial p expressed by Bernstein–Bézier patch over D, (d) $\overline{p}$ is the best linear approximant of p with respect to L_2 norm over domain D. 72

43 One iteration of `bilinclip` II: (e) p and $\overline{p}$ are both expressed in Bernstein-Bézier form, the error bound δ^p is obtained as the maximum length of the thick black vertical bars, (f) the lower and upper bounds $p^L = \overline{p} - \delta^p$ and $p^U = \overline{p} + \delta^p$, (g) boundaries of polynomials p and q are intersected with plane xy in order to construct planar bounding strips P and Q, respectively, (h) the intersection of strips is bounded by the least square that defines the new domain D'. 73

44 Construction of the new domain: (a) Parallelogram P is bounded by the least rectangle of which sides are parallel with axes x and y, (b) intersection of the original domain D and the rectangle R gives the new domain D'. 74

45 The null set of p and q in unit domain 75

46 Polynomials p and q over unit domain. 76

47 Astroid and Maltese Cross curve on domain $\langle -5, 5\rangle \times \langle -5, 5\rangle$. 78

48 Descartes leaf and Lemniscat of Bernoulli on $\langle -5, 5\rangle \times \langle -5, 5\rangle$ 79

49 Intersection of curves of bidegree $(6, 6)$ 80

50 Intersection of curves of bidegree $(8, 8)$ 81

51 Single root of bivariate polynomial system. 82

List of Tables

1 Scheme of point construction via de Casteljau algorithm. . . . 14

2 Convergence rates of the algorithms `quadclip` and `bezclip`. . 54

3 Number of operations per step of the iteration for various values of the degree n. 54

4 Time per iterations in microseconds for various degrees n. . . 55

5 Example 6 (single root): Number of iterations N and computing time t in μs for various values of degree n and accuracy ε. The times for more than 16 significant digits (shown in italic) have been obtained by extrapolation. 58

6 Example 7 (double root): Number of iterations N and computing time t in μs for various values of degree n and accuracy ε. The times for more than 16 significant digits (shown in italic) have been obtained by extrapolation. 61

7 Example 8 (near double root): Number of iterations N and computing time t in μs for various values of degree n and accuracy ε. The times for more than 16 significant digits (shown in italic) have been obtained by extrapolation. 64

8 Example 11; computing time, number of all iterations and number of roots within prescribed accuracy ε. 77

9 Example 12; number of iterations with respect to the accuracy ε. 78

10 Example 13; computing time and number of iterations within prescribed accuracy ε 79

11 Example 14; time and number of iterations within prescribed accuracy ε 80

12 Example 15; time and number of iterations within prescribed accuracy ε 82

References

Ahn, Y. J., B.-G. Lee, Y. Park, and J. Yoo (2004), Constrained polynomial degree reduction in the L_2-norm equals best weighted Euclidean approximation of Bézier coefficients, *Comput. Aided Geom. Design* **21**, 181-191.

Efremov, A., V. Havran and H.-P. Seidel (2005), Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces, in *Proc. 21st Spring Conference on Computer Graphics*, ACM Press, New York, 127–135.

Ciesielski, Z. (1987), The basis of B-splines in the space of algebraic polynomials, *Ukrainian Math. J.* **38**, 311–315.

Eck, M. (1992), Degree reduction of Bézier curves, *Comp. Aided Geom. Design* **10**, 237–251.

Gautschi, W. (1997), Numerical analysis, Birkhäuser, Boston, MA.

Elber, G., and M.-S. Kim (2001), Geometric constraint solver using multivariate rational spline functions, In *The Sixth ACM/IEEE Symposium on Solid Modeling and Applications*, Ann Arbor, MI, 1–10.

Gatellier, G., A. Labrouzy, B. Mourrain, and J. Técourt (2003), Computing the topology of three-dimensional algebraic curves, In: T. Dokken and B. Jüttler, editors, *Computational methods for algebraic spline sufaces (COMPASS)*, Springer, Berlin, 27–43.

Gonzalez-Vega, L., and I. Necula, Efficient topology determination of implicitly defined algebraic plane curves, *Comp. Aided Geom. Design* **19**, 719–743.

Kim, M.-S., G. Elber and J.-K. Seong, Geometric computations in parameter space, in *Proc. 21st spring conference on Computer graphics*, ACM Press, New York, 27–32.

Hoschek, J. and D. Lasser (1993), *Fundamentals of computer aided geometric design*. AK Peters, Wellesley, MA.

Jüttler B. (1998), The dual basis functions of the Bernstein polynomials. *Adv. Comput. Math.* **8**, 345–352.

Ko, K., T. Sakkalis, and N. Patrikalakis (2005). Resolution of multiple roots of nonlinear polynomial systems, *Int. J. Shape Model.*, **11.1**, 121–147.

Lee, K. (1999), *Principles of CAD/CAM/CAE systems*. Prentice Hall.

Li, T. (2003), Numerical solution of polynomial systems by homotopy continuation methods. In F. Cucker, editor, *Special Volume: Foundations of Computational Mathematics*, volume XI of *Handbook of Numerical Analysis*, North–Holland, 209–304.

Lutterkort, D., J. Peters and U. Reif (1999), Polynomial degree reduction in the L_2-norm equals best Euclidean approximation of Bézier coefficients. *Comput. Aided Geom. Design* **16**, 607-612.

Mourrain, B., and J.-P. Pavone (2005), Subdivision methods for solving polynomial equations, Technical Report no. 5658, INRIA Sophia Antipolis, `http://www.inria.fr/rrrt/rr-5658.html`.

Nishita, T., T. Sederberg, and M. Kakimoto (1990), Ray tracing trimmed rational surface patches. *Siggraph Proceedings*, ACM, 337–345.

Nishita, T., and T. W. Sederberg (1990), Curve Intersection using Bézier Clipping. *Computer–Aided Design* **22.9** 538–549.

Patrikalakis, N. M., and T. Maekawa (2002a), Chapter 25: Intersection problems, in G. Farin, J. Hoschek, and M.-S. Kim, editors, *Handbook of computer aided geometric design.* Amsterdam: Elsevier, 623–649.

Patrikalakis, N. M., and T. Maekawa (2002b), *Shape interrogation for computer aided design and manufacturing,* Springer, Berlin.

Sherbrooke, E. C., and N. M. Patrikalakis (1993), Computation of the solutions of nonlinear polynomial systems, *Comput. Aided Geom. Design* **10**, 379–405.

Sommese, A. J., and C. W. Wampler (2005), *The numerical solution of systems of polynomials arising in engineering and science.* World Scientific.

Sunwoo, H. (2005), Matrix representation for multi-degree reduction of Bézier curves. *Comput. Aided Geom. Design* **22**, 261-273.

Wang, H., J. Kearney, and K. Atkinson (2003), Robust and efficient computation of the closest point on a spline curve, in T. Lyche et al., editors, *Curve and surface design: Saint Malo 2002*, Nashboro Press, Brentwood, 397–405.