Computing roots of polynomials by quadratic clipping

Michael Bartoň[†,1] and Bert Jüttler[‡,1]

[†] *Johann Radon Institute for Computational and Applied Mathematics, Austria*
[‡] *Johannes Kepler University Linz, Institute of Applied Geometry, Austria*

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.

Key words: root finding, polynomial, Bézier clipping

1 Introduction

Efficient and robust algorithms which compute the solutions (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).

[1] Both authors were supported by the Austrian Science Fund (FWF) through the SFB F013 “Numerical and Symbolic Scientific Computing”, subproject 15.

Various geometric problems, 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 are guaranteed to find *all* solutions. Two major approaches exist:

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. 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 2.3. 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 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, which is 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 limitations of CAD systems.

The remainder of this paper is organized as follows. In the first section, we give information about the root finding problem, about degree reduction and about the classical technique of Bézier clipping. Section 2 describes the new algorithm and analyzes its order of convergence. In Section 3, we provide a detailed comparison of the new algorithm with Bézier clipping with respect to criteria such as computational effort, rate of convergence and computing times. Finally we conclude this paper and discuss future work, including the generalization to the multivariate case.

2 Preliminaries

After formulating the root–finding problem, we present a simple technique for degree reduction. Finally we describe the classical technique of Bézier clipping for isolating roots of univariate polynomials.

2.1 The root–finding problem

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

$$B_i^n(t) = \binom{n}{i} \frac{(t-\alpha)^i(\beta-t)^{n-i}}{(\beta-\alpha)^n} \tag{1}$$

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,

$$p(t) = \sum_{i=0}^{n} b_i B_i^n(t), \quad t \in [\alpha,\beta], \tag{2}$$

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 unit interval $[0,1]$. All roots of p within the unit interval 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.

2.2 Degree reduction and dual basis

The process of approximating a polynomial p 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 inner product

$$\langle f,g\rangle^{[\alpha,\beta]} = \int_\alpha^\beta f(t)\,g(t)\,\mathrm{d}t \tag{3}$$

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

$$\|f\|_2^{[\alpha,\beta]} = \frac{1}{h}\sqrt{\langle f, f\rangle^{[\alpha,\beta]}}, \tag{4}$$

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. More precisely, for any affine transformation

$$\mathcal{A} : t \mapsto A_0 + A_1\, t \tag{5}$$

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,

$$\|f\|_2^{[\alpha,\beta]} = \|f \circ \mathcal{A}^{-1}\|_2^{\mathcal{A}([\alpha,\beta])}. \tag{6}$$

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.,

$$q = \arg\min_{q\in\Pi^k} \|p - q\|_2^{[\alpha,\beta]}. \tag{7}$$

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

$$\langle B_i^k, D_j^k\rangle^{[\alpha,\beta]} = \delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}, \quad i, j = 0 \ldots k. \tag{8}$$

The polynomials D_j^k can be represented with respect to the Bernstein basis,

$$D_i^k(t) = \frac{1}{h}\sum_{j=0}^{k} c_{i,j}\, B_j^k(t), \quad i = 0, \ldots, k, \tag{9}$$

with the coefficients

$$c_{p,q} = \frac{(-1)^{p+q}}{\binom{k}{p}\binom{k}{q}} \sum_{j=0}^{\min(p,q)} (2j+1)\binom{k+j+1}{k-p}\binom{k-j}{k-p}\binom{k+j+1}{k-q}\binom{k-j}{k-q} \tag{10}$$

which have been derived by 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 (2) and (7)) with respect to the interval $[\alpha, \beta]$ may be computed from

$$q(t) = \sum_{j=0}^{k} \langle p(t), D_j^k(t) \rangle^{[\alpha,\beta]} \, B_j^k(t) = \sum_{j=0}^{k} \left(\sum_{i=0}^{n} b_i \beta_{i,j}^{n,k} \right) B_j^k(t), \tag{11}$$

with the coefficients

$$\beta_{i,j}^{n,k} = \langle B_i^n(t), D_j^k(t) \rangle^{[\alpha,\beta]}. \tag{12}$$

Using the identity

$$\langle B_i^m, B_j^n \rangle^{[\alpha,\beta]} = h \frac{\binom{m}{i}\binom{n}{j}}{(m+n+1)\binom{m+n}{i+j}}, \tag{13}$$

these coefficients can be computed from (9) and (10). Note that these coefficients do not depend on the interval $[\alpha, \beta]$, since the factors h in (9) and (13) cancel each other.

Example 1 The degree reduction coefficients for $n = 5$ and $k = 2$ form the matrix

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

The coefficients of q are obtained by multiplying the row vector $(b_0, \ldots, b_5)$ of the coefficients of p by this matrix.

2.3 Bézier clipping and its convergence rate

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 curve with control points

$$\mathbf{b}_i = (\frac{(n-i)\alpha + i\beta}{n}, b_i), \quad i = 0, \ldots, n. \tag{15}$$

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

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

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

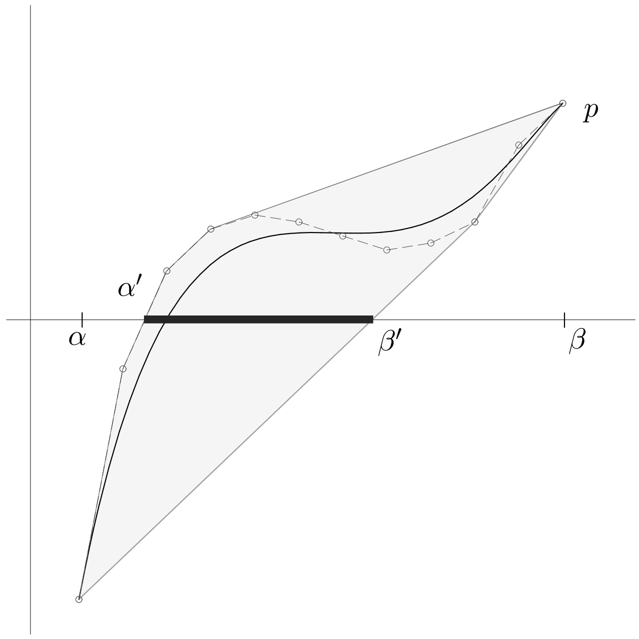


Fig. 1. Bézier clipping: The next interval containing the roots is obtained by intersecting the convex hull of the control polygon with the t–axis.

contained in the interval which is obtained by intersecting $\mathcal{C}$ with the t–axis. This observation, which is illustrated by Figure 1, 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 degree reduction

We describe a new algorithm for isolating the roots and analyze its convergence rates in the cases of roots with multiplicities 1 and 2.

3.1 Algorithm

Based on degree reduction to a quadratic polynomial ($k = 2$), we propose a new technique for computing the roots, see Algorithm 2 (`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. 2 (left). 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

$$\|p - q\|_{\infty}^{[\alpha,\beta]} = \max_{t\in[\alpha,\beta]} |p(t) - q(t)|, \tag{16}$$

  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 2. The bound is chosen as

$$\delta = \max_{i=0,\ldots,n} |b_i - c_i|, \tag{17}$$

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

1: **if** length of interval $[\alpha, \beta] \geq \varepsilon$ **then**
2: $q \leftarrow$ generate a quadratic polynomial by applying degree reduction with respect to the L^2 inner product on $[\alpha, \beta]$ to p.
3: $\delta \leftarrow$ compute bound on $\|p - q\|_{\infty}^{[\alpha,\beta]}$ by comparing the Bernstein–Bézier representations of p and q.
4: $m \leftarrow q - \delta$ {lower bound}
5: $M \leftarrow q + \delta$ {upper bound}
6: **if** the strip enclosed by m, M does not intersect the t–axis within $[\alpha, \beta]$ **then**
7: **return** $(\emptyset)$
8: **else**
9: 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 or 2.
10: **if** $\max_{i=1,\ldots,k} |\alpha_i - \beta_i| > \frac{1}{2}|\alpha - \beta|$ **then**
11: **return** (`quadclip` $(p,[\alpha, \frac{1}{2}(\alpha + \beta)])$ $\cup$ `quadclip` $(p,[\frac{1}{2}(\alpha + \beta), \beta])$).
12: **else**
13: $S \leftarrow \emptyset$
14: **for** $i = 1, \ldots, k$ **do**
15: $S \leftarrow S \cup$ `quadclip`$(p, [\alpha_i, \beta_i])$
16: **end for**
17: **return** (S)
18: **end if**
19: **end if**
20: **else**
21: **return** $([\alpha, \beta])$
22: **end if**

see Fig. 2 (left), 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.

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

$$\forall t \in [\alpha, \beta] : \quad m(t) \leq p(t) \leq M(t). \tag{18}$$

- In lines 6–19 we analyze the strip enclosed by m and M and its intersection with the t–axis, see Fig. 2, right. 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 3.
- 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 $[0, 1]$, the call `quadclip`$(p, [0, 1])$ returns an interval containing that root. Similar to Bézier clipping, quadratic clipping may

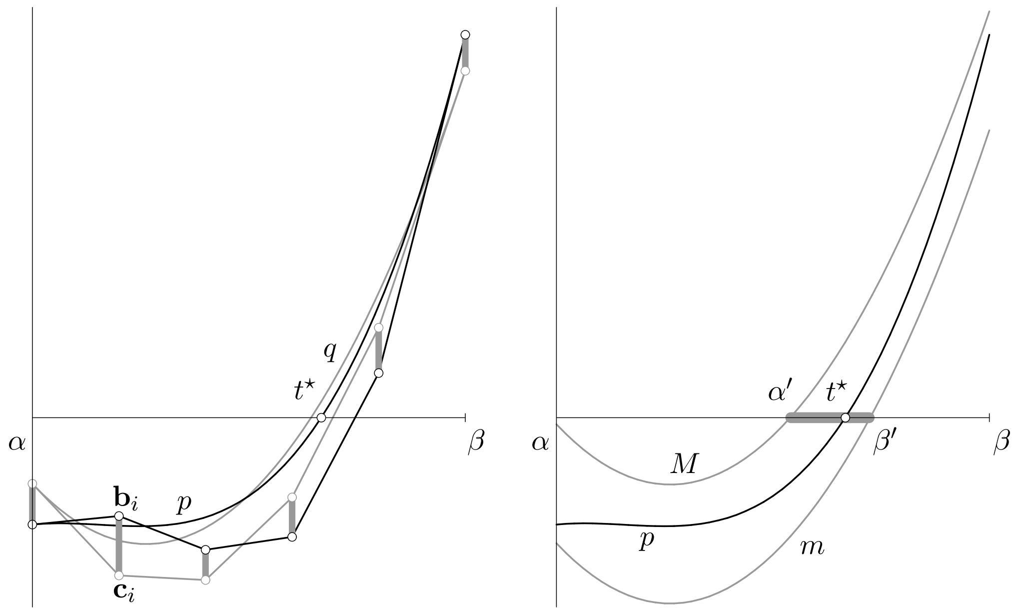


Fig. 2. One iteration of `quadclip` in the case of a single root. Left: the polynomial p and its quadratic approximation q, along with the control polygons. The error bound δ is obtained as the maximum length of the thick grey vertical bars. Right: the lower and upper bounds $m = q - \delta$ and $M = q + \delta$. The intersection of the strip enclosed by them with the t–axis defines the new interval $[\alpha', \beta']$.

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

$$(\beta_{i,j}^{2,5})_{i=1,\dots,2;j=0,\dots,5} = \begin{bmatrix} 1 & \frac{3}{5} & \frac{3}{10} & \frac{1}{10} & 0 & 0 \\ 0 & \frac{2}{5} & \frac{3}{5} & \frac{3}{5} & \frac{2}{5} & 0 \\ 0 & 0 & \frac{1}{10} & \frac{3}{10} & \frac{3}{5} & 1 \end{bmatrix}. \tag{19}$$

Remark 3 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

$$\tau_{1|2} = \frac{d_1 - d_0 \pm \sqrt{D}}{d_2 - 2d_1 + d_0}. \tag{20}$$

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 (20) becomes numerically unstable. In this situation we apply Bézier clipping to the control polygon of g in order to bound the roots.

3.2 Convergence rate

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

Lemma 4 *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. Due to the equivalence of norms in finite–dimensional real linear spaces, there exist constants C_1 and C_2 such that

$$\forall r \in \Pi^n : \quad \|r\|_{\mathrm{BB},\infty}^{[\alpha,\beta]} \leq C_1 \|r\|_2^{[\alpha,\beta]} \quad \text{and} \quad \|r\|_2^{[\alpha,\beta]} \leq C_2 \|r\|_\infty^{[\alpha,\beta]}, \tag{21}$$

where the three norms are the maximum (ℓ_∞) norm of the Bernstein–Bézier coefficients, the L^2 norm (4) and the maximum norm

$$\|r\|_\infty^{[\alpha,\beta]} = \max_{t \in [\alpha,\beta]} |r(t)|, \tag{22}$$

all with respect to the interval $[\alpha, \beta]$. The constants C_1 and C_2 do not depend on the given interval $[\alpha, \beta]$, since all three norms are invariant with respect to affine transformations of the t–axis; cf. (5) and (6).

Consequently,

$$\begin{aligned}\delta = \|p - q\|_{\mathrm{BB},\infty}^{[\alpha,\beta]} \leq C_1 \|p - q\|_2^{[\alpha,\beta]} \leq C_1 \|p - Q_\alpha\|_2^{[\alpha,\beta]} \leq \\ \leq C_1 C_2 \|p - Q_\alpha\|_\infty^{[\alpha,\beta]} \leq \frac{1}{6}\, C_1 C_2 \max_{t_0 \in [0,1]} |p'''(t_0)|\ h^3,\end{aligned} \tag{23}$$

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

Lemma 5 *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*

$$\|p - q\|_\infty^{[\alpha,\beta]} \leq V_p\, h^3, \quad \|p' - q'\|_\infty^{[\alpha,\beta]} \leq D_p\, h^2, \quad \textit{and} \ \|p'' - q''\|_\infty^{[\alpha,\beta]} \leq A_p\, h, \tag{24}$$

with $h = \beta - \alpha$, where $\|.\|_\infty^{[\alpha,\beta]}$ is defined as in (22).

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

$$\|r\|_\star^{[\alpha,\beta]} = \|r\|_\infty^{[\alpha,\beta]} + h\, \|r'\|_\infty^{[\alpha,\beta]} + h^2\, \|r''\|_\infty^{[\alpha,\beta]}, \tag{25}$$

satisfies

$$\|r\|_\star^{[\alpha,\beta]} \le C_3 \|r\|_2^{[\alpha,\beta]} \tag{26}$$

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,

$$\begin{aligned} \|p-q\|_\star^{[\alpha,\beta]} &= \|p-q\|_\infty^{[\alpha,\beta]} + h\,\|p'-q'\|_\infty^{[\alpha,\beta]} + h^2\,\|p''-q''\|_\infty^{[\alpha,\beta]} \le \\ &\le C_3\|p-q\|_2^{[\alpha,\beta]} \le C_3\|p-Q_\alpha\|_2^{[\alpha,\beta]} \le C_2C_3\|p-Q_\alpha\|_\infty^{[\alpha,\beta]} \le \\ &\le \frac{1}{6}\,C_2C_3 \max_{t_0\in[0,1]} |p'''(t_0)|\,h^3, \end{aligned} \tag{27}$$

where Q_α is the quadratic Taylor polynomial at $t=\alpha$ to p. Clearly, this implies (24) $\square$.

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 6 *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. The call `quadclip`$(p,[0,1])$ generates a sequence of intervals

$$([\alpha_i,\beta_i])_{i=0,1,2,\ldots} \tag{28}$$

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 5, the inequalities

$$\|p'-p'(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{4}p'(t^\star) \quad \text{and} \quad \|q_i'-p'\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{4}p'(t^\star) \tag{29}$$

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

$$\|q_i'-p'(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \le \frac{1}{2}\,p'(t^\star), \quad \text{hence} \quad \forall t\in[\alpha_i,\beta_i]:\ q_i'(t) > \frac{1}{2}\,p'(t^\star). \tag{30}$$

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 4. Thus, the lengths h_i of the intervals

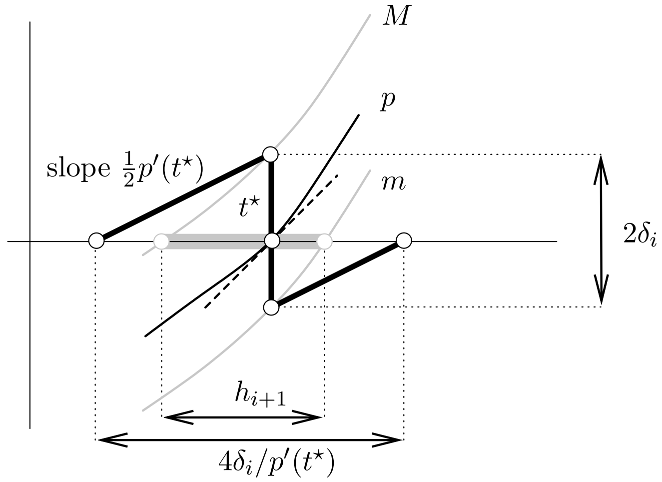


Fig. 3. Proof of Eq. (31)

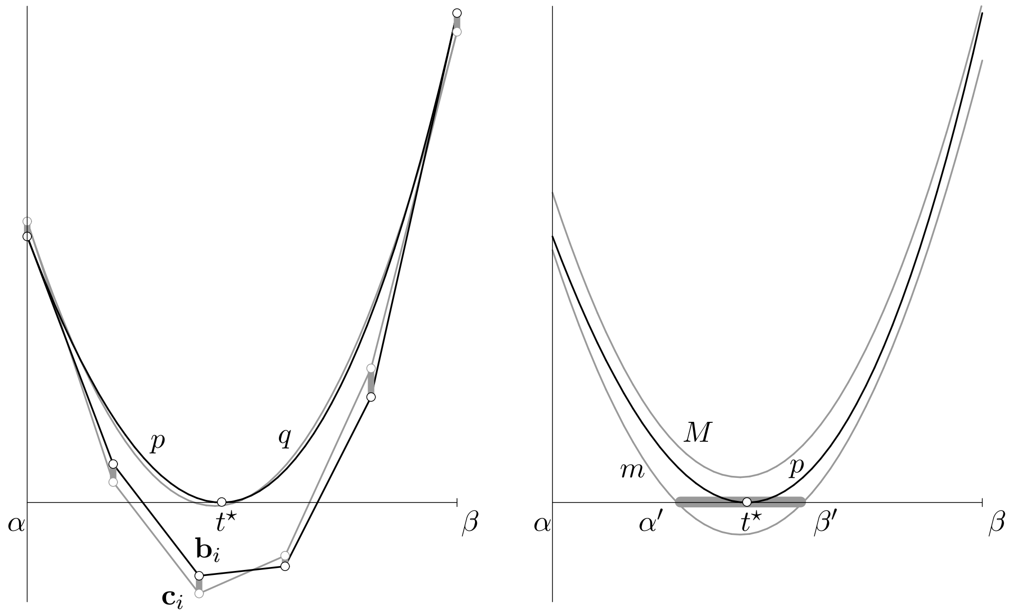


Fig. 4. One iteration of `quadclip` in the case of a double root. See caption of Figure 2.

satisfy

$$h_{i+1} \leq \frac{4\,C_p}{p'(t^\star)}\,h_i^3 \tag{31}$$

for all but finitely many values of i, see Fig. 3. $\square$

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 4 for an illustration.

Theorem 7 *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 gener-*

ated by `quadclip` *which contain that root has the convergence rate* $d = \frac{3}{2}$.

Proof. Similar to the proof of the previous Theorem, we analyze the sequence (28) of intervals with lengths h_i generated by the algorithm which 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 5, the inequalities

$$\|p'' - p''(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \leq \frac{1}{4}p''(t^\star) \quad \text{and} \quad \|q_i'' - p''\|_\infty^{[\alpha_i,\beta_i]} \leq \frac{1}{4}p''(t^\star) \tag{32}$$

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

$$\|q_i'' - p''(t^\star)\|_\infty^{[\alpha_i,\beta_i]} \leq \frac{1}{2}p''(t^\star), \quad \text{hence} \quad \forall t \in [\alpha_i, \beta_i] : q_i''(t) > \frac{1}{2}p''(t^\star). \tag{33}$$

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

$$m_i = \frac{a_i}{2}(t - t^\star)^2 + b_i(t - t^\star) + c_i \tag{34}$$

with certain real coefficients $a_i = q_i''(t^\star)$, $b_i = q_i'(t^\star)$ and c_i. According to (33), the leading coefficient satisfies

$$a_i \geq \frac{1}{2}p''(t^\star) \tag{35}$$

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

$$|b_i| = |p'(t^\star) - q'(t^\star)| \leq \|p' - q'\|_\infty^{[\alpha_i,\beta_i]} \leq D_p\, h_i^2 \tag{36}$$

and

$$\begin{aligned} |c_i| &= |p(t^\star) - m(t^\star)| \leq |p(t^\star) - q(t^\star)| + |q(t^\star) - m(t^\star)| \\ &\leq \|p - q\|_\infty^{[\alpha_i,\beta_i]} + \delta_i \leq (V_p + C_p)\, h_i^3. \end{aligned} \tag{37}$$

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

Table 1
Convergence rates of the algorithms `quadclip` and `bezclip`.

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

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

$$h_{i+1} \leq 2\sqrt{\frac{b_i^2}{a_i^2} - \frac{2c_i}{a_i}} \leq \frac{|b_i|}{a_i} + \sqrt{\frac{2|c_i|}{a_i}} \leq \frac{2D_p}{p''(t^\star)} h_i^2 + \sqrt{\frac{4(C_p + V_p)}{p''(t^\star)}}\, h_i^{3/2}. \quad (38)$$

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

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 1.

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 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 2 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 2) and that this interval has shrunk by more than $\frac{1}{2}$, cf. line 5 of Algorithm 1 and line 10 of Algorithm 2. Also, the number of operations needed for computing

Table 2
Number of operations per step of the iteration for various values of the degree n.

degree n	quadclip $\pm$	$*\div$	$\leq$	$\sqrt{}$	$\lvert\cdot\rvert$	$\sum$	bezclip $\pm$	$*\div$	$\leq$	$\sqrt{}$	$\lvert\cdot\rvert$	$\sum$
2	120	75	30	4	0	229	90	30	5	0	0	125
4	228	115	32	4	2	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
Time per iterations in microseconds for various degrees n.

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

the convex hull for Algorithm 1 varies slightly; here we assume to have a convex control polygon, since this is the limit case in general.

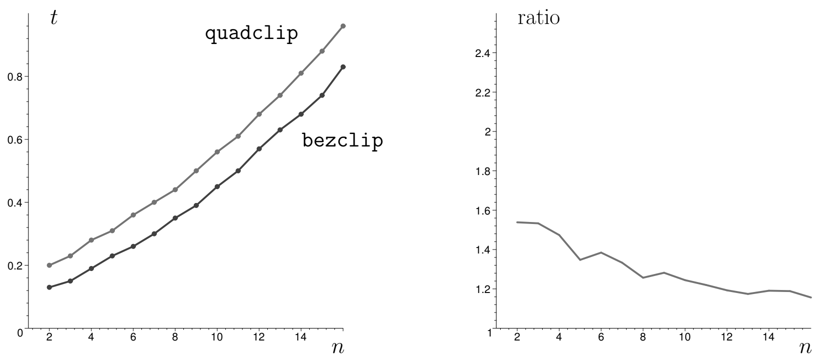
The classical Bézier clipping has a slight advantage, though the computational costs of both methods are roughly comparable. The number of operations grows *quadratically* with the degree n. For both algorithms, the computational effort grows linearly, except for the quadratic grow caused by de Casteljau's 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) 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 3.

In addition, Fig. 5a shows the relation between computing times and polynomial degree, and Fig. 5b 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



(a) (b)

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

root").

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

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

in order to compute the single root $\frac{1}{3}$ in the interval $[0, 1]$. Table 4 reports the number of iterations and the computing times for various values of the 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 3). In addition, Figure 6 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 9 (Double root) We applied the algorithms `bezclip` and `quadclip` to the four polynomials

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

Table 4
Example 8 (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.

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*

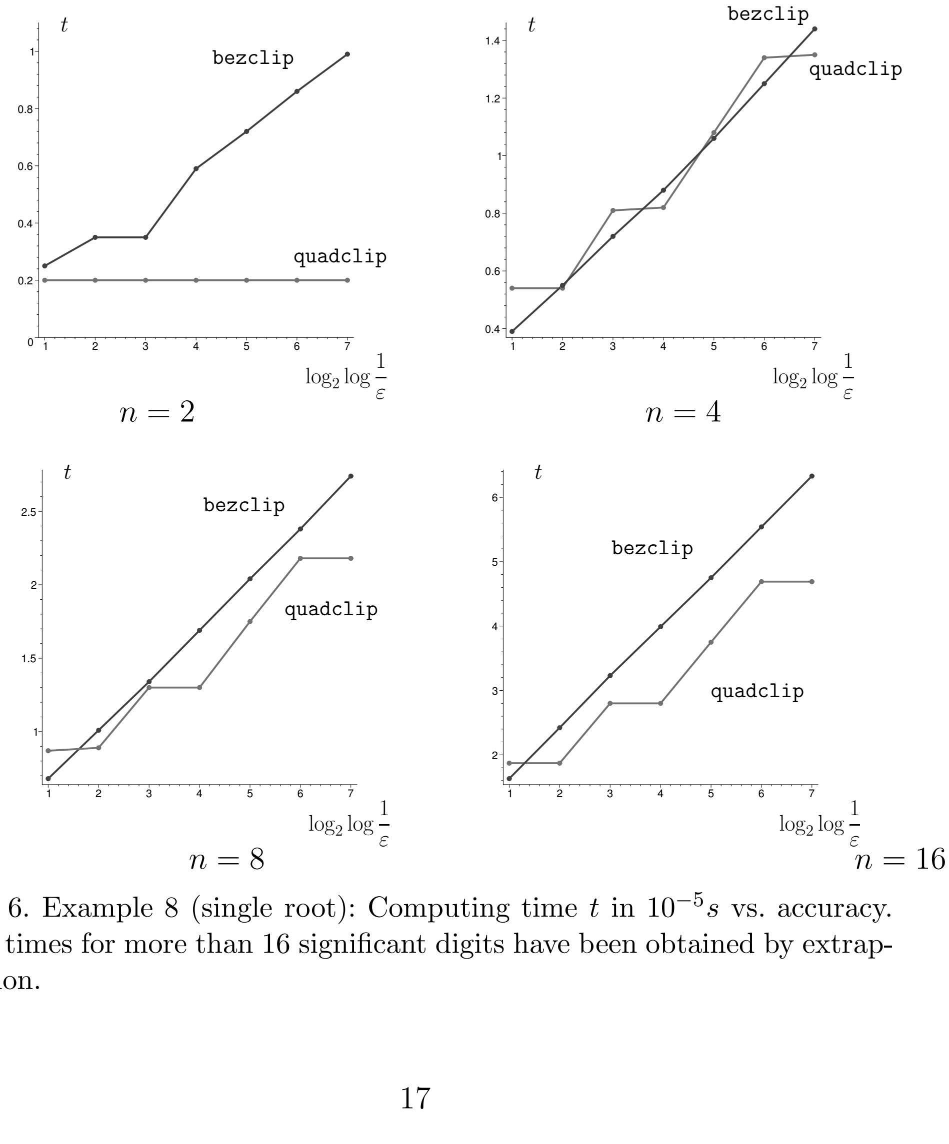
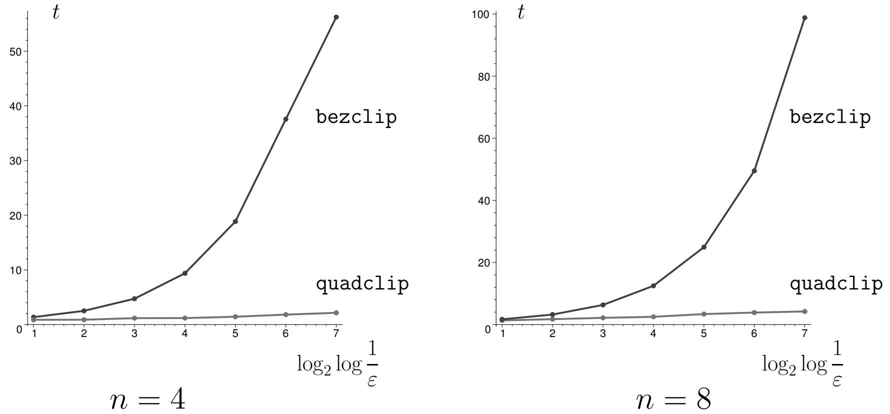


Fig. 6. Example 8 (single root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation.

Table 5
Example 9 (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.

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*



$n = 4$ $n = 8$

Fig. 7. Example 9 (double root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation.

in order to compute the double root $\frac{1}{2}$ in the interval $[0, 1]$. Table 5 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 3). In addition, Figure 7 visualizes the relation between computing times and desired accuracy.

For these four polynomials, the new algorithm (`quadclip`) performs far better than Bézier clipping. This is due to the higher convergence rate (1.5) of the new algorithm.

Table 6
Example 10 (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.

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*

Example 10 (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 6 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 3). In addition, Figure 7 visualizes the relation between computing times and desired accuracy.

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).

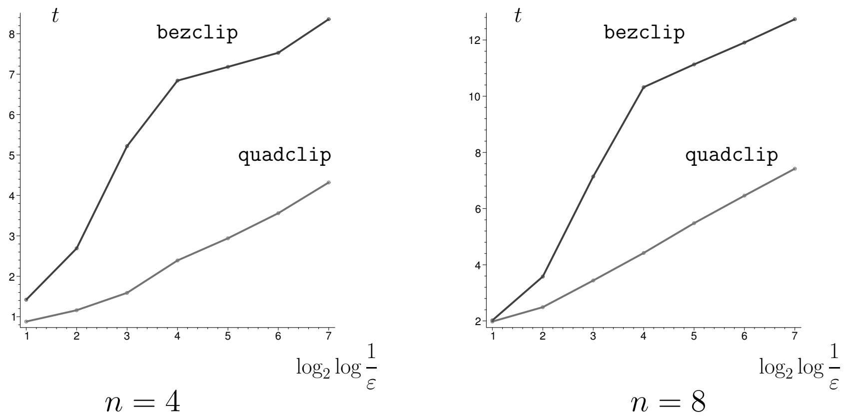


Fig. 8. Example 10 (near double root): Computing time t in $10^{-5}s$ vs. accuracy. The times for more than 16 significant digits have been obtained by extrapolation.

5 Concluding remarks

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 generalization 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.

Our future work will focus on the extension of the technique to the multivariate case. In a first step, we will formulate an algorithm which is based on approximation by linear polynomials. We expect that this leads to an algorithm with quadratic convergence for single roots, leading to results comparable to those of Mourrain and Pavone (2005). In a second step, we are working on several ideas which are expected to lead to faster convergence for single roots and superlinear convergence for double roots.

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 Clip-

ping. *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.