The Complexity of an Adaptive Subdivision Method for Approximating Real Curves

Michael Burr, Shuhong Gao, Elias Tsigaridas

The Complexity of an Adaptive Subdivision Method for Approximating Real Curves*

Michael A. Burr†
Clemson University
Clemson, South Carolina
burr2@clemson.edu

Shuhong Gao‡
Clemson University
Clemson, South Carolina
sgao@clemson.edu

Elias Tsigaridas§
Sorbonne Universités, UPMC
Univ Paris 06, CNRS, INRIA,
Laboratoire d'Informatique de
Paris 6 (LIP6), Équipe PolSys,
4 place Jussieu, 75252 Paris
Cedex 05, France
elias.tsigaridas@inria.fr

ABSTRACT

We present the first complexity analysis of the algorithm by Plantinga and Vegter for approximating real implicit curves and surfaces. This approximation algorithm certifies the topological correctness of the output using both subdivision and interval arithmetic. In practice, it has been seen to be quite efficient; our goal is to quantify this efficiency.

We focus on the subdivision step (and not the approximation step) of the Plantinga and Vegter algorithm. We begin by extending the subdivision step to arbitrary dimensions. We provide *a priori* worst-case bounds on the complexity of this algorithm both in terms of the number of subregions constructed and the bit complexity for the construction. Then, we use continuous amortization to derive adaptive bounds on the complexity of the subdivided region. We also provide examples showing our bounds are tight.

*This work was significantly advanced while the first and third authors were supported at the Fields Workshop on Algebra, Geometry, and Proof in Symbolic Computation.

†Partially supported by a grant from the Simons Foundation (#282399 to Michael Burr) and National Science Foundation Grant CCF-1527193.

‡Partially supported by the National Science Foundation under Grants CCF-1407623, DMS-1403062 and DMS-1547399.

§Partially supported by HPAC (ANR ANR-11-BS02-013) and an FP7 Marie Curie Career Integration Grant.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

1. INTRODUCTION

Subdivision-based algorithms are one of the most commonly used algorithms in many fields, from computational geometry and graphics to solving of polynomials and mathematical programming [22, 18, 28, 2, 23, 39, 16, 1].

Subdivision-based algorithms are intrinsically adaptive. Starting with the domain of interest, usually an axis-aligned box, they recursively split it into sub-domains, eliminating those that do not contain a solution or interesting features of the problem at hand. At the end, the algorithm produces a union of sub-domains (boxes) which lie in the initial domain.

The main advantages of subdivision-based algorithms are their great flexibility and their local nature. Because of their recursive character, they are easy to implement, and this makes them popular among practitioners. Moreover, they are often efficient in practice because they only perform additional subdivisions near difficult features. It is exactly because of these advantages, however, that the complexity analysis of subdivision-based algorithms is particularly challenging. To analyze them we need to understand, in depth, the local complexity of the input instance and how the inclusion/exclusion predicates behave on them.

For univariate problems, the analysis of subdivision algorithms is well-understood, and there are several results, especially for the case of approximating the roots of polynomials, e.g., [38, 27]. However, for higher dimensions, very little is known. For example, there are no explicit complexity results for pure subdivision-based algorithms for approximating curves and surfaces.

In this paper we fill this gap. We consider the following problem: Plantinga and Vegter [25] presented a subdivision-based algorithm for correctly approximating curves and surfaces, see Figure 1. We call this the `PV` algorithm. It takes, as input, a polynomial $f \in \mathbb{R}[x, y]$ or $\mathbb{R}[x, y, z]$, whose real zero set is bounded and smooth, and a region $I \subseteq \mathbb{R}^2$ or $\mathbb{R}^3$. From this input data, it constructs a piecewise-linear approximation to the zero set of f in I. The approximation has the correct topology in the sense that there is an ambient isotopy between the approximation and the zero set; additionally, by further subdivisions, the Hausdorff distance between the approximation and the zero set can be made as small as desired. The authors of [25] claim that the `PV` algorithm is efficient in practice, but, to our knowledge, there is no prior complexity analysis of the `PV` algorithm.

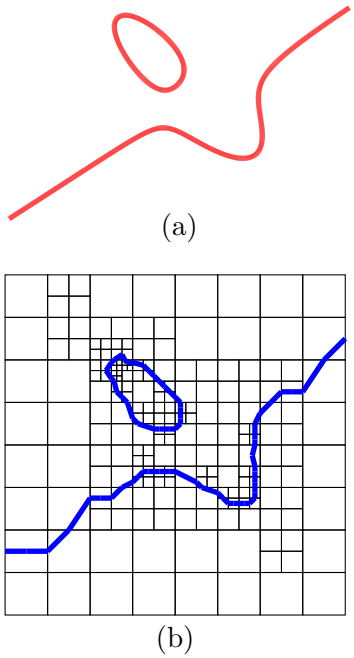

Figure 1: (a) The graph of the real trace of the curve $f = 3y^3+3xy^2-2x^3-3y^2+xy+3x^2-3y+3x+2$. (b) The approximation produced by the PV algorithm [25] as well as the boxes constructed by the algorithm.

We provide the first complexity analysis for approximating curves and surfaces using the PV algorithm. Our bounds exploit local point estimates, called *local size bounds*, to quantify the amount of work needed by the algorithm at each point of the input domain. For the case of curves, we prove a worst-case bit complexity bound of $2^{\widetilde{O}(d^3\tau)}$. Additionally, we extend the predicates for the PV algorithm to all dimensions and analyze the complexity of these tests in two and higher dimensions. Moreover, we provide an adaptive bound on the size of the subdivision using *continuous amortization*, first developed in [8] and extended to higher dimensions in [6]. Our results are the first application of continuous amortization to a pure high dimensional problem. Furthermore, we prove that our bounds are tight in Lemma 6.1.

1.1 Related Work

The design of efficient subdivision-based algorithms that are *output-sensitive*, *precision-sensitive*, *certified*, and exploit the underlying *structure* of the problem is a great challenge and an active area of research. A big step in this direction is the introduction of *soft tests* [34, 38] that, roughly speaking, replace hard exact tests, usually comparisons with zero, with approximate computations, and they are exact in the limit. They introduce a new notion of correctness called resolution-exactness. In this context, it is exactly the continuous amortization tool [6, 8] that captures the complexity of the soft predicates. Therefore, continuous amortization is a key tool for the analysis of such algorithms.

The previous work on subdivision methods and exclusion/inclusion predicates is quite extensive and so we only scratch its surface. For works that focus on classical exclusion/exclusion algorithms but without bit complexity bounds we refer the reader to [17, 35, 14]. For other approaches for approximating curves and surfaces we refer the reader to [11, 4, 3, 10] and the references therein. Recently, there is an extension to the case of analytic functions [17, 38]. For the problem of isolating the roots of polynomials we refer the reader to [23, 20, 19, 12, 36, 24, 7, 27, 9] and the references therein. There are also approaches [24] that achieve locally quadratic convergence towards the simple roots of polynomial systems and there very efficient in practice. Another interesting direction of subdivision algorithms, of more geometric nature, concerns the approximation of algebraic varieties [30, 25, 5, 29, 37, 21], and the computation of the approximate Voronoi diagrams [39]. There are also quite important applications of these algorithms to the problem of robot motion planning [33].

1.2 Notation

We use $\texttt{dist}_{\mathbb{C}}(x, f)$ for the distance, in $\mathbb{C}$, of the point x to the hypersurface defined by the polynomial f. The point x might be real or complex. We also use $\texttt{Bdist}_a(V, W)$ to indicate that we are interested in the minimum distance between $V, W \subset \mathbb{C}^n$ inside a hyperbox of $\mathbb{C}^n$ whose corners have bit-size at most a.

For an interval or a (hyper-)box J we denote the midpoint as $m = m(J)$ and side length as $w = w(J)$. The notation $\Box f(J)$ denotes the interval (over-)approximation of f applied to the region J. In this paper, because of its nice properties, we use the centered interval form for $\Box f$ which is based on the Taylor expansion of f at the midpoint of J, see Remark 2.2 or [26] for further details.

We use $O(\cdot)$ and $O_B(\cdot)$ to denote the arithmetic complexity and bit complexity, respectively. The soft-O notation, $\widetilde{O}(\cdot)$ and $\widetilde{O}_B(\cdot)$, mean that we are ignoring logarithmic factors. By $V_{\mathbb{C}}(f)$, $V_{\mathbb{R}}(f)$, or $V_I(f)$ we denote the zero set of a polynomial f over the complex numbers, real numbers, or the region I, respectively.

1.3 Outline

The rest of the paper is organized as follows:

In Section 2, we recall the PV algorithms and the inclusion and inclusion predicates that it uses. We unify and generalize these predicates to make them applicable to higher dimensions. In Section 3, we derive the local size bounds for the inclusion/exclusion predicates for the PV algorithms. In Section 4, we use the local size bounds to provide worst-case estimates on the number of boxes and the bit complexity of the PV algorithms in terms of the size of the input. In Section 5, we present an introduction to and apply continuous amortization to present adaptive complexity bounds for the PV algorithms. Finally, in Section 6 we present example to demonstrate the tightness of our bounds.

2. THE MODIFIED PLANTINGA AND VEGTER ALGORITHM

In this section, we present the subdivision part of the PV algorithm. We give details on how PV exploits interval arithmetic to achieve certified computations. We slightly modify the algorithm to unify the tests and to make the subdivisions applicable in arbitrary dimensions.

2.1 The PV Algorithm [25]

Let $f \in \mathbb{R}[x,y]$ or $\mathbb{R}[x,y,z]$ be a square-free polynomial such that its real zero set $V_{\mathbb{R}}(f)$ is smooth and bounded. The PV algorithm recursively subdivides an initial bounding square or cube I for the variety with a quad-tree or oct-tree data structure until at least one of the following two tests holds on each subregion J. In the literature, these tests are often referred to as C_0 and C_1:

$$C_0(J) := 0 \notin \Box f(J) \quad C_1(J) := 0 \notin \langle \Box \nabla f(J), \Box \nabla f(J) \rangle.$$

When $C_0(J)$ holds, the variety does not enter the region J and so J can be discarded. On the other hand, when $C_1(J)$ holds, the curve or surface does not bend much within the region J. More precisely, given f and I, the PV algorithms construct a partition P of I so that C_0 or C_1 is true on each region in P. Initially, $P = \{I\}$:

Algorithm 2.1. Main subdivision of PV algorithm

Repeatedly subdivide (into 4 or 8 children) each $J \in P$ until one of the following conditions hold:

$C_0(J)$ is TRUE or $C_1(J)$ is TRUE.

After every sub-region J satisfies $C_0(J)$ or $C_1(J)$, the authors of [25] perform post-processing steps, which include balancing the tree, evaluating the sign of f on the corners of each J in P, and using sign changes along the sides of regions J to detect and approximate the curve or surface. This approximation is topologically correct as there is an ambient isotopy between the approximation and the variety $V_{\mathbb{R}}(f)$. Additionally, by further subdivision, the isotopy can be made sufficiently small so that the Hausdorff distance between the approximation and the variety is as small as desired. We note that it is possible to extend the PV algorithm in the plane to provide an approximation even when $V_{\mathbb{R}}(f)$ is unbounded, $V_{\mathbb{R}}(f)$ is singular, and I is not a bounding box, see [5]. In this paper, however, we focus on the original PV algorithm without the restriction of a bounded curve.

2.2 The Subdivisions of the PV Algorithm

Our main focus is on computing the number of regions that the PV algorithms construct, see Algorithm 2.1, and not on the approximation of the curve or surface, per se. Therefore, we focus exclusively on the C_0 and C_1 tests and apply them in arbitrary dimensions. More precisely, let $f \in \mathbb{R}[x_1, \cdots, x_n]$ be such that its real zero set $V_{\mathbb{R}}(f)$ is smooth. Let $I \subseteq \mathbb{R}^n$ be a n-dimensional real cube. Then, we can generalize the tests C_0 and C_1, along with Algorithm 2.1, to n dimensions, where the subdivision splits an n-cube into 2^n children. However, we mention that, in this case, we no longer use the output of the algorithm to construct an approximation to $V_{\mathbb{R}}(f)$.

2.3 Extending the C_1 test

The predicate $C_1(J)$ has the following two consequences that are fundamental in the proof of correctness of the PV algorithm in [25]: (1) If a region J satisfies the conditions, then, in J, there cannot be any pair of gradient vectors which are orthogonal to each other. (2) The variety $V_{\mathbb{R}}(f)$ is parametrizable in the direction of at least one of the coordinate axes. Fact (2) is a direct consequence of Fact (1), but it is used so frequently in the proofs in [25], that it is worthwhile to mention it explicitly.

We now modify and extend test C_1 so that C_0 and C_1 have the same form. Let the function $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, defined as $g(x_1, \cdots, x_n, y_1, \cdots, y_n) = \langle \nabla f(x_1, \cdots, x_n), \nabla f(y_1, \cdots, y_n) \rangle$. It follows that for a region J, if $0 \notin \Box g(J \times J)$, then there is no pair of gradient vectors in J which are orthogonal to each other. We use this reformulation because it implies the same two consequences, Facts (1) and (2), as the original C_1 test, but the application of interval arithmetic appears as the last step as opposed to an intermediate step. In particular, both tests C_0 and C_1 are of the same type, i.e., they consist of testing where 0 appears in the interval formulation of a function applied to a region. For the rest of the paper, all references to the C_1 test refer to this new C_1 test.

REMARK 2.2. *We use the centered interval form for interval arithmetic, see [26], which is based on the Taylor expansion of the polynomial at the midpoint of the region. In particular, let J be the n-cube with midpoint $m = m(J)$ and side length $w = w(J)$. Then, test C_0 simplifies to*

$$(C_0) \qquad |f(m)| > \sum_{|\alpha| \geq 1} \frac{|\partial^\alpha f(m)|}{\alpha!} \left(\frac{w}{2}\right)^{|\alpha|}$$

where $\alpha = (\alpha_1, \cdots, \alpha_n) \in \mathbb{N}^n$ is a multi-index. In particular, $|\alpha| = \sum \alpha_i$ is the sum of the entries in α, $\partial^\alpha = \partial_1^{\alpha_1} \cdots \partial_n^{\alpha_n}$ is the operator of partial derivatives applied with multiplicity determined by the entries of α, and $\alpha! = \prod \alpha_i!$ is the product of the factorials of the entries in α.

Since the C_1 test is based on the function g whose domain is $2n$-dimensional and the square $J \times J$ has midpoint (m, m) and the same side length as J, the C_1 test becomes

$$(C_1) \quad |\nabla f(m)|^2 > \sum_{|\alpha|+|\beta| \geq 1} \left| \sum_{i=1}^{n} \frac{\partial^{\alpha+e_i} f(m) \cdot \partial^{\beta+e_i} f(m)}{\alpha! \cdot \beta!} \right| \left(\frac{w}{2}\right)^{|\alpha|+|\beta|}$$

where e_i is the i-th standard basis vector, and $\alpha, \beta \in \mathbb{N}^n$ are multi-indices.

3. LOCAL SIZE BOUND FOR THE PV ALGORITHM

The key to the bounds in this paper is a function, called a *local size bound*, on $\mathbb{R}^n$ which locally describes the maximum amount of work that is required at each point in $\mathbb{R}^n$ (cf [6]):

DEFINITION 3.1. *Let C be a predicate on n-dimensional cubes. A* local size bound *for C is a function $F : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ with the property that*

$$F(x) \leq \inf_{\substack{J,\ n\text{-dim cube} \\ J \ni x \\ C(J)=\text{FALSE}}} \text{Vol}_n(J).$$

In other words, $F(x)$ is a lower bound on the n-dimensional volume of a n-dimensional cube which contains x, but fails the stopping criterion test.

We develop a local size bound for the PV algorithm. For the PV algorithm, the local size bound relates the value of the partial derivatives at a point in $\mathbb{R}^n$ to the distance from that point to the variety. The goal is to derive point estimates for the C_0 and C_1 predicates, Corollaries 3.6 and 3.7. As our first step, we reduce a higher-dimensional problem to a collection of one-dimensional problems as follows:

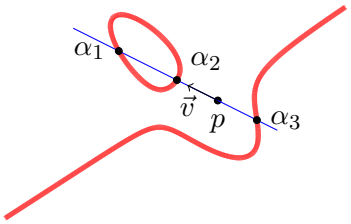

Figure 2: For a polynomial $f \in \mathbb{R}[x,y]$ and a point $p \in \mathbb{R}^2$, we consider the roots of f, $\alpha_1, \alpha_2, \alpha_3$, in the direction of a unit vector v.

DEFINITION 3.2. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$, $p \in \mathbb{R}^n$, and $v \in S^{n-1}$. Define $f_v(t)$ to be the univariate polynomial passing through p and in the direction v, i.e., $f_v(t) = f(p + tv)$, see Figure 2. Then, define Σ_{f_v} be the sum of the reciprocals of the complex roots of f_v, i.e.,*

$$\Sigma_{f_v}(p) = \sum_{s \in V(f_v)} \frac{1}{|s|}.$$

As in [7], $\Sigma_{f_v}(p)$ links the Taylor coefficients of f to the geometry of the zero set of f. This relationship is explicitly explored in the following two results.

LEMMA 3.3. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$, $p \in \mathbb{R}^n$, and $v \in S^{n-1}$. Then*

$$\left| \frac{1}{f(p)} \cdot \frac{d^k f(p+tv)}{dt^k} \bigg|_{t=0} \right| \leq (\Sigma_{f_v}(p))^k \leq \left(\frac{\deg(f)}{\texttt{dist}_{\mathbb{C}}(p,f)} \right)^k.$$

PROOF. The claim is trivial when $k = 0$. Since f_v is a univariate polynomial, the first inequality follows directly from [7, Lemma 2.1]. The second inequality follows because, in the sum for $\Sigma_{f_v}(p)$, there are at most $\deg(f)$ terms and each element of the sum is the inverse of the distance between p and a point on $V_{\mathbb{C}}(f)$, which is bounded above by the inverse of the distance to the closest point. □

The interval approximations in the PV predicates are based on the centered form, which, in turn is based on the Taylor expansion of the input polynomial, see Remark 2.2. We, therefore, use the geometric bounds from Lemma 3.3 to bound the Taylor coefficients in terms of the geometry of the zero set of f. The proof of Proposition 3.4 and Corollary 3.5 as well as other results that we need to effectively bound the local size bound appear in the full version of the paper.

PROPOSITION 3.4. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$ and $p \in \mathbb{R}^n$. Then, for all multi-indices $\alpha \in \mathbb{N}^n$,*

$$\left| \frac{1}{f(p)} \binom{k}{\alpha} \frac{\partial^{|\alpha|} f}{\partial x^\alpha}(p) \right| \leq 2^{(n-1)(|\alpha|+1)} \left(\frac{\deg(f)}{\texttt{dist}_{\mathbb{C}}(p,f)} \right)^{|\alpha|}$$

where $\binom{k}{\alpha} = \frac{k!}{\alpha!}$ is the multinomial coefficient.

Each of the predicates for the PV algorithm test whether 0 is included in an interval approximation to the value of an appropriate function on a region. In practice, this test is performed by comparing the sizes of the Taylor coefficients and the size of the input region to the value of the polynomial at a given point, see Remark 2.2. In particular, if the inequality appearing in the conclusion of the following corollary holds, then 0 is not included in the interval approximation, for additional details, see [7].

COROLLARY 3.5. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$ and $p \in \mathbb{R}^n$. Suppose that $0 < w \leq \frac{\texttt{dist}_{\mathbb{C}}(p,f) \ln(1+2^{2-2n})}{2^{n-1}\deg(f)}$. Then*

$$\left| \sum_{k=1}^{\deg(f)} \sum_{|\alpha|=k} \frac{1}{k!} \binom{k}{\alpha} \frac{1}{f(p)} \cdot \frac{\partial^{|\alpha|} f}{\partial x^\alpha}(p) \left(\frac{w}{2}\right)^k \right| < 1.$$

We now develop bounds on the size of J which guarantee the success of the PV predicates which can be applied to any point within J, not merely the midpoint.

COROLLARY 3.6. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$ and $J \subseteq \mathbb{R}^n$. Assume that there is a point $x \in J$ such that*

$$w(J) \leq \frac{2\ln\left(1+2^{2-2n}\right) \texttt{dist}_{\mathbb{C}}(x,f)}{2^n \deg(f) + \sqrt{n}\ln\left(1+2^{2-2n}\right)}.$$

Then, $C_0(J)$ is true.

PROOF. This result follows from Corollary 3.6 and [7, Section 3]. □

COROLLARY 3.7. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$ and $g \in [x_1, \cdots, x_n, y_1, \cdots, y_n]$ where $g(x_1, \cdots, x_n, y_1, \cdots, y_n) = \langle \nabla f(x_1, \cdots, x_n), \nabla f(y_1, \cdots, y_n) \rangle$. Let $J \subseteq \mathbb{R}^n$, and suppose that there is a point $(a,b) \in J \times J$ such that*

$$w(J) \leq \frac{2\ln\left(1+2^{2-4n}\right) \texttt{dist}_{\mathbb{C}}((a,b),g)}{2^{2n+1}(\deg(f)-1) + \sqrt{2n}\ln\left(1+2^{2-4n}\right)}.$$

Then, $C_1(J)$ is true.

PROOF. This result follows from Corollary 3.6 and [7, Section 3]. □

4. WORST-CASE BOUNDS

Observe that Corollary 3.6 applies to a polynomial in n variables and Corollary 3.7 uses a polynomial in $2n$ variables. Therefore, the varieties $V_{\mathbb{C}}(f)$ and $V_{\mathbb{C}}(g)$ are in different dimensional spaces. In the arguments that follow, it is easier to study both of these varieties in the same dimensional space. Therefore, we consider $V_{\mathbb{C}}(f)$ as a subset of the diagonal in $2n$-dimensional space. In particular, let the variables of $\mathbb{C}^{2n}$ be $\{x_1, \cdots, x_n, y_1, \cdots, y_n\}$; the diagonal Δ consists of the points of the form $x_i = y_i$. Then, Δ is n-dimensional and we identify $\mathbb{C}^n$ with Δ, in particular, we write $V^{\Delta}_{\mathbb{C}}(f)$ for the collection of all points $(x_1, \cdots, x_n, y_1, \cdots, y_n)$ such that $f(x_1, \cdots, x_n) = 0$ and $x_i = y_i$ for all i.

4.1 The Global Depth of the Subdivision Tree

We prove a global bound on the depth of the subdivision tree.

PROPOSITION 4.1. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$, and let $g \in \mathbb{R}[x_1, \cdots, x_n, y_1, \cdots, y_n]$ where $g(x_1, \cdots, x_n, y_1, \cdots, y_n) = \langle \nabla f(x_1, \cdots, x_n), \nabla f(y_1, \cdots, y_n) \rangle$. Suppose that $I \subseteq \mathbb{R}^n$ and $V_I(f)$ is smooth, i.e., or all $x \in I$, $f(x)$ and $g(x,x)$ are not both zero. In particular, let*

$$0 < \delta \leq \min_{x \in I} \max\{\texttt{dist}_{\mathbb{C}}(x,f), \texttt{dist}_{\mathbb{C}}((x,x),g)\}.$$

Define

$$D = \max\left\{ \frac{2\ln\left(1+2^{2-2n}\right)}{2^n\deg(f) + \sqrt{n}\ln\left(1+2^{2-2n}\right)}, \frac{2\ln\left(1+2^{2-4n}\right)}{2^{2n+1}(\deg(f)-1) + \sqrt{2n}\ln\left(1+2^{2-4n}\right)} \right\}.$$

The PV *algorithm performs at most* $\left(\frac{2w(I)}{D\delta}\right)^n$ *subdivisions.*

PROOF. $\frac{1}{2}D\delta$ is a lower bound on the width of a region which occurs as a leaf in the subdivision tree. Therefore, we can divide the total n-dimensional volume into regions of this size. □

Let $f \in \mathbb{R}[x_1, \cdots, x_n]$ define a smooth variety, and let $g \in \mathbb{R}[x_1, \cdots, x_n, y_1, \cdots, y_n]$ where $g(x_1, \cdots, x_n, y_1, \cdots, y_n) = \langle \nabla f(x_1, \cdots, x_n), \nabla f(y_1, \cdots, y_n) \rangle$. Since f is smooth as a complex variety, there do not exist points $x \in \mathbb{R}^n$ such that $x \in V_{\mathbb{C}}(f)$ and $(x, x) \in V_{\mathbb{C}}(g)$. Suppose that ε is a separation bound between $V_{\mathbb{C}}^{\Delta}(f)$ and $V_{\mathbb{C}}(g)$ and bewteen $\mathbb{R}^n$ and $V_{\mathbb{C}}(f, \nabla f \cdot \nabla f)$. Then, for all $x \in \mathbb{R}^n$, either the distance $\mathtt{dist}_{\mathbb{C}}(x, f)$ is at least $\frac{\varepsilon}{2\sqrt{2}}$ or the distance $\mathtt{dist}_{\mathbb{C}}((x, x), g)$ is at least $\frac{\varepsilon}{2}$. Therefore, we may let $\delta = \frac{\varepsilon}{2\sqrt{2}}$ in Proposition 4.1.

4.2 Bounds on the distance between varieties

Suppose that the input polynomial f has integer coefficients. Then, we can express the bound in Proposition 4.1 in terms of the degree and the maximum bitsize of the coefficients of the input polynomial. For this, we need to bound ε, that appears in the previous section, from below. In other words, we need to bound the distance between $V_{\mathbb{C}}^{\Delta}(f)$ and $V_{\mathbb{C}}(g)$. We restrict our attention to the interior of an n-dimensional cube such that a is an upper bound on the bitsize of the corners of the box. Following [31], we define the following distance function, for $a > 0$,

$$\begin{aligned}&\mathtt{Bdist}_a(V, W)\\ &\quad = \inf\{\|p - q\| \,:\, p \in V, q \in W, \lg(\|p\|), \lg(\|q\|) \le a\}\end{aligned}$$

We observe that $\mathtt{Bdist}_a(V, W)$ is based on the distance between complex varieties, and it is not restricted to their real portions. The following theorem presents a lower bound on the distance of two varieties, when at least one of them is defined as a complete intersection, and is due to Martin Sombra, see also [31, §2.3.3], [13].

THEOREM 4.2. *Let V be of pure dimension r and W be of dimension s and defined by $f_1, \ldots, f_{n-s} \in \mathbb{Z}[x_1, \ldots, x_n]$. Let $d = \max_i\{\deg(f_i)\}$ and the maximum logarithmic height of the polynomials f_i be $h = \max_i\{h(f_i)\}$. We can bound the distance between V and W within a ball $B(0, 2^a)$ as*

$$\begin{aligned}&-\lg \mathtt{Bdist}_a(V, W) \le\\ &\quad d^{n-s}(h(V) + \deg(V)\left((n-s)\frac{h}{d} + (4r+10)\lg(n+2) + 2\right),\end{aligned}$$

where $\deg(V)$ and $h(V)$ denote the degree and the canonical height of V, respectively.

Furthermore, if we assume that V is a complete intersection defined by polynomials $g_1, \ldots, g_{n-r}$, such that $\deg(g_i) \le d_2$ and $h(g_i) \le h_2$, then

$$h(V) \le (n-r)\, d_2^{n-r}\left(\frac{h_2}{d_2} + \lg(n+1)\right). \tag{1}$$

In this case the distance between V and W becomes:

$$\lg \Delta \ge -c(n) d^{n-s} d_2^{n-r}\left(\frac{h}{d} + \frac{h_2}{d_2} + a + 1\right), \tag{2}$$

where $c(n)$ is a constant depending only on n.

To use Theorem 4.2 and Equation (2) to bound the distance between $V_{\mathbb{C}}^{\Delta}(f)$ and $V_{\mathbb{C}}(g)$, we consider the hypersurface V defined by the following equation:

$$g(x_1, \cdots, x_n, y_1, \cdots, y_n) = \nabla f(x_1, \cdots, x_n) \cdot \nabla f(y_1, \cdots, y_n).$$

Moreover, W is defined by the system of equations

$$f(x_1, \cdots, x_n) \qquad \text{and} \qquad \{y_i = x_i\}.$$

Since g is a single equation, V is a $r = (2n-1)$-dimensional complex hypersurface. On the other hand, f defines an $s = (n-1)$-dimensional complex variety. Both varieties are complete intersections.

We assumed that the input polynomial f has integer coefficients, that is $f \in \mathbb{Z}[x_1, \cdots, x_n]$ such that $d = \deg(f)$ is the degree of f and that the maximum bitsize of the coefficients, aka logarithmic height, is $h(f) = h_2 = \tau$. Then, $\deg(W) = d$, $\deg(\nabla f) \le d - 1$, and $h(\nabla f) = \tau + \lg(d)$. From this, it follows that $h_1 = h(g) = O(\tau + \lg(nd))$ and $\delta_1 = \deg(V) = \deg(g) \le 2d - 2$.

We can use the bound of Eq. (1) to bound the (logarithmic) height of a variety, and in our case V, as

$$h(V) \le 2\tau + (4n+2)\lg d = O(\tau + n\lg(nd)).$$

Combining these inequalities with Theorem 4.2, see also Equation (2), we obtain the following bound for the distance between $V_{\mathbb{C}}^{\Delta}(f)$ and $V_{\mathbb{C}}(g)$ within an n-dimensional cube with coordinates having bitsize bounded by a:

$$\begin{aligned}&-\lg(\mathtt{Bdist}_a(V_{\mathbb{C}}^{\Delta}(f), V_{\mathbb{C}}(g))\\ &\qquad \le d^{n+1}(6n\tau + 40nd\lg(nd) + 2da)\\ &\qquad = O(d^{n+1}\,(n\tau + nd\lg(nd) + da)).\end{aligned}$$

The previous bound bounds ϵ and thus δ in Proposition 4.1. Therefore, we obtain the following explicit bound for the number of steps, or in other words the number of boxes, of the PV algorithm.

THEOREM 4.3. *The number of steps that* PV *algorithm performs is at most*

$$2^{O(nd^{n+1}(n\tau + nd\lg(nd) + 9n + d)a)}.$$

where a is the maximum bitsize of the coordinates of the corners of I.

4.3 Overall complexity bound

The bound on the number of steps that we have computed in Theorem 4.3 is an important quantity needed for the goal of estimating the overall bit complexity of the PV algorithm. PV is a subdivision algorithm that, roughly speaking, mimic the process of binary search. Therefore, to bound its Boolean complexity we have to multiply the number steps (subdivisions) that it performs with the worst case Boolean complexity of each step. Theorem 4.3 provides the number of steps. It remains to estimate the complexity of each step. A closer look at the predicates needed for the realization of the algorithm reveals that each step of the PV algorithm consists of a multivariate Taylor shift. Given a polynomial $F \in \mathbb{Z}[x_1, \ldots, x_n]$ and numbers $a_1, \ldots, a_n$, we need to compute the coefficients of $F(x_1 + a_1, \ldots, x_n + a_n)$.

We perform this operation recursively. First, we consider the bivariate case. We are given a polynomial $F \in \mathbb{Z}[x_1, x_2]$ of total degree d and maximum coefficient bitsize τ, and

integers a_1, and a_2 of bitsize ϱ. We want to estimate the complexity of computing the polynomial $F(x_1+a_1, x_2+a_2)$.

We consider the polynomial as $F \in (\mathbb{Z}[x_1])[x_2]$, that is as a univariate polynomial in x_2 with coefficients in x_1; we write it as $F = \sum_{i=0}^{d} F_i(x_1)\, x_2^i$. The Taylor shift is $F(x_1 + a_1, x_2 + a_2) = \sum_{i=0}^{d} F_i(x_1 + a_1)\,(x_2 + a_2)^i$. There are $d+1$ univariate Taylor shifts, $F_i(x_1+a_1)$, that we have to perform. Each costs $\widetilde{O}_B(d^2\varrho + d\tau)$ [32]; hence the cost of all of them is $\widetilde{O}_B(d^3\varrho + d^2\tau)$. We also have to compute (expand) the $d+1$ polynomials $(x_2+a_2)^i$. Each "expansion" costs at most $\widetilde{O}_B(d^2\varrho)$, as it corresponds, in the worst case, to $O(\lg d)$ multiplications of two polynomials of degree at most d and bitsize $\widetilde{O}(d\varrho)$. We perform all of them in $\widetilde{O}_B(d^3\varrho)$.

Finally, to get $F(x_1+a_1, x_2+a_2) = \sum_{i=0}^{d} F_i(x_1 + a_1)\,(x_2+a_2)^i$, we need to perform the $d+1$ multiplications between polynomials $F_i(x_1 + a_1)$ and $(x_2 + a_2)^i$ to obtain each term in the sum, and $d+1$ additions. The multiplications dominate the complexity. As each multiplication corresponds, in the worst case, to the multiplication of two polynomials of degree d and bitsize $\widetilde{O}(\tau + d\varrho)$ and $\widetilde{O}(d\varrho)$, it costs $\widetilde{O}_B(d\tau + d^2\varrho)$. All of them cost $\widetilde{O}_B(d^3\varrho + d^2\tau)$.

LEMMA 4.4. *The Taylor shift for a bivariate polynomial of total degree d and maximum coefficient bitsize τ costs $\widetilde{O}_B(d^3\varrho + d^2\tau)$.*

Using induction we obtain the following result:

COROLLARY 4.5. *The Taylor shift for a multivariate polynomial of total degree d in n variables and maximum coefficient bitsize τ costs $\widetilde{O}_B(d^{n+1}\varrho + d^n\tau)$.*

To obtain the worst case complexity of each step of the PV algorithm, we replace ϱ in Corollary 4.5 with the (exponent of the) bound of Theorem 4.3. To see this, notice that at each step of the subdivision we increase by 1 the number of bits of the number we have to perform Taylor shifts with. Hence, in the worst case, we have to perform a Taylor shift with number have as bitsize the number of subdivisions. For the overall complexity of the algorithm we multiply this bound with the number of steps of Theorem 4.3. In the worst case, since we construct a polynomial of degree $2d-2$ in $2n$ variables, each multivariate Taylor shift costs $\widetilde{O}_B(n^2d^{2n+3} + nd^{2n+2}(\tau + da))$ and the overall complexity of the algorithm is

$$2^{d^{n+1}(6n^2\tau + 42n^2 d\lg(nd) + 2(d+n)a) + n}\widetilde{O}_B(n^2d^2(\tau + d(1+a))),$$

where a is the maximum bitsize of the coordinates of the corners of I.

For the $2D$ case, $n = 2$, that commonly appears in applications, we obtain the following bound for applying PV is

THEOREM 4.6. *The bit complexity of the PV algorithm for curves, is $\widetilde{O}_B(2^{d^3(24\tau + 168\lg(d) + 2da + 1)})$.*

5. ADAPTIVE BOUNDS

Continuous amortization was introduced in [8] as a way to adaptively analyze the complexity of subdivision-based algorithms. The theory of continuous amortization was extended to higher dimensions in [6]. We recall this technique in the special case for computing the number of regions created by a subdivision algorithm in $\mathbb{R}^n$ which recursively subdivides n-dimensional cubes in $\mathbb{R}^n$ into 2^n identical smaller n-dimensional cubes.

THEOREM 5.1 ([8, 6]). *Let F be a local size bound (see Definition 3.1) for a stopping criterion C and I be an n-dimensional cube. The number of regions formed by a subdivision algorithm which recursively subdivides n-dimensional cubes in $\mathbb{R}^n$ into 2^n identical smaller n-dimensional cubes is at most*

$$\max\left\{1, \int_I \frac{2^n dV_n}{F(x)}\right\}$$

where dV_n is the n-dimensional volume form. If the algorithm does not terminate, then the integral is infinite.

We can use continuous amortization to express the complexity of the PV algorithm. We express the PV subdivision as a subdivision of $2n$-dimensional real space, but restrict our attention to the diagonal Δ. Since the real part of the diagonal Δ can be identified with $\mathbb{R}^n$ via projection onto the first n coordinates, we can interpret all of our subdivisions as subdivisions of $\mathbb{R}^n$. In particular, for $J \times J \in \mathbb{R}^{2n}$, under a standard subdivision, this region would be split into 2^{2n} children. For our case, however, we only need to consider the subregions which intersect the diagonal; in other words, we only consider subregions of the form $L \times L$. By via the identification to $\mathbb{R}^n$ this subdivision corresponds to the standard subdivision in $\mathbb{R}^n$ where L is a child of J, and J is 2^n times bigger than L.

PROPOSITION 5.2. *Let $f \in \mathbb{R}[x_1, \cdots, x_n]$, and let $g \in \mathbb{R}[x_1, \cdots, x_n, y_1, \cdots, y_n]$ where $g(x_1, \cdots, x_n, y_1, \cdots, y_n) = \langle \nabla f(x_1, \cdots, x_n), \nabla f(y_1, \cdots, y_n)\rangle$. Suppose that $I \subseteq \mathbb{R}^n$. The number of regions after the subdivision performed by the PV algorithm (before balancing) is bounded above by the maximum of 1 and*

$$2^n \int_I \min\left\{ \left(\frac{2^n \deg(f) + \sqrt{n}\ln\left(1 + 2^{2-2n}\right)}{2\ln\left(1 + 2^{2-2n}\right)\mathtt{dist}_{\mathbb{C}}(x, f)}\right)^n, \left(\frac{2^{2n+1}(\deg(f) - 1) + \sqrt{2n}\ln\left(1 + 2^{2-4n}\right)}{2\ln\left(1 + 2^{2-4n}\right)\mathtt{dist}_{\mathbb{C}}((x,x), g)}\right)^n \right\} dV_n$$

where dV_n is the n-dimensional volume form.

PROOF. These results follow almost directly from a standard application of continuous amortization, see [6]. The only difference is that the argmax in the proof of continuous amortization must be taken over the diagonal, $(J \times J) \cap \Delta = \{(a, a) : a \in J\}$ instead of all of $J \times J$. □

This integral provides a more adaptive and accurate estimate on the complexity than the worst-case *a priori* bounds based on the size of the input. Moreover, this integral can be evaluated even when the input polynomial has complex (but not real) singularities. Examples of the evaluation of this integral appear in the remainder of this paper.

6. EXAMPLES

Both bounds are exponential with respect to the degree of the polynomial f and the number of variables. They remain exponential even if we assume that the number of variables is

constant. In [25], the authors show that for several examples the computation time is efficient in practice. The following lemma, which is also illustrated in Figure 3(a), proves that this exponential behavior is optimal, up to constants in the exponents.

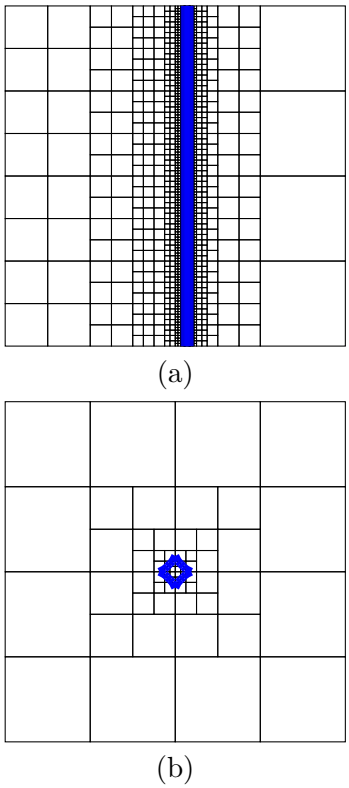

Figure 3: (a) The output of the PV algorithm for $f(x,y) = (x^n - 2(ax-1)^2)(x^n - (ax-1)^2)$. The solutions to $f(x,y) = 0$ are close vertical lines ($n = a = 3$). The width of boxes between vertical lines is at most $2a^{-\frac{n}{2}-1}$ and they extend the entire length of the initial region. The number of regions is bounded by: $\Omega(w(I)a^{\frac{n}{2}+1})$ where $w(I)$ is the width of he initial region. (b) The approximation of $f(x,y) = x^2 + y^2 + \varepsilon^2$. The number of regions is bounded by: $O(\lg(w(I)) - \lg(\varepsilon))$.

LEMMA 6.1. *The bound of Thm. 4.3 is asymptotically tight.*

PROOF. Following the construction in [15], consider the Mignotte polynomial $P(x) = x^d - 2(ax-1)^2$ and the related polynomial $P_2(x) = x^d - (ax-1)^2$ where a is a sufficiently large positive integer. The product $P(x)P_2(x)$ is of degree $2d$ and the largest coefficient is of size $2a^4$. In [15], it is shown that the product $P(x)P_2(x)$ has (at least) three roots in the interval $(a^{-1} - h, a^{-1} + h)$ where $h = a^{-d/2-1}$. Treating $P(x)P_2(x)$ as a polynomial in n variables, we see that the PV algorithm to approximate the variety in an n-dimensional cube I of side length $w(I)$ requires subdividing to regions of side length at most $2h$ to separate the three vertical hyperplanes in the interval $(a^{-1} - h, a^{-1} + h)$. Since this occurs along an entire hyperplane of the input region, the number of small boxes is, at least, $\frac{w(I)^{n-1}}{2h} = \frac{1}{2}w(I)^{n-1}a^{d/2+1}$, which is exponential in both the size of the input of the box and the size of the coefficients of the polynomial. □

Even though our bounds are optimal, in practice, these are quite pessimistic, as the actual separation bounds do not follow the worst case behavior, see Figure 3(b). This is illustrated in the following two examples:

EXAMPLE 6.2. *Fix $\varepsilon > 0$ and consider $f(x_1,x_2) = x_1^2 + x_2^2 + \varepsilon^2$. Then, $\mathtt{dist}_{\mathbb{C}}((x_1,x_2),f) = \sqrt{\frac{x_1^2+x_2^2}{2} + \varepsilon^2}$ and $\mathtt{dist}_{\mathbb{C}}((x_1,x_2,x_1,x_2),g) = \sqrt{x_1^2+x_2^2}$. For any square I, the number of regions constructed by the PV algorithm is $O(\lg(w(I)) - \lg(\varepsilon))$.*

EXAMPLE 6.3. *Fix $\varepsilon > 0$ and consider $f(x_1,x_2) = x_1^2 + x_2^2 - \varepsilon^2$. Then,*

$$\mathtt{dist}_{\mathbb{C}}((x_1,x_2),f) = \begin{cases} |\sqrt{x_1^2+x_2^2} - \varepsilon| & x_1^2+x_2^2 \le 4\varepsilon^2 \\ \sqrt{\frac{x_1^2+x_2^2}{2} - \varepsilon^2} & x_1^2+x_2^2 > 4\varepsilon^2 \end{cases} \text{ and}$$

$\mathtt{dist}_{\mathbb{C}}((x_1,x_2,x_1,x_2),g) = \sqrt{x_1^2+x_2^2}$. *For any square I, the number of regions constructed by the PV algorithm is $O(\lg(w(I)) - \lg(\varepsilon))$.*

Moreover, for each of these examples, the minimum distance between $V_{\mathbb{C}}^{\Delta}(f)$ and $V_{\mathbb{C}}(g)$ is at most ε. Therefore, a bound coming from Proposition 4.1 would be much larger than the bound continuous amortization provides.

It remains an open question to deduce the bit complexity bounds for the PV algorithms in the general case from the adaptive bounds of Proposition 5.2. Since the complexity of the algorithm can be exponential in the inputs, the integral must be described in terms of additional geometric and intrinsic parameters.

Acknowledgments. The authors are greatful to Martin Sombra for his comments and for providing them the lower bound on the distance between varieties.

7. REFERENCES

[1] E. Allgower, M. Erdmann, and K. Georg. On the complexity of exclusion algorithms for optimization. *Journal of Complexity*, 18(2):573–588, 2002.

[2] I. Babuvška and W. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[3] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, and G. Vegter. Meshing of surfaces. In *Effective Computational Geometry for Curves and Surfaces*, pages 181–229. Springer, 2006.

[4] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surface meshing. *Discrete & Computational Geometry*, 1(39):138–157, 2008.

[5] M. Burr, S. W. Choi, B. Galehouse, and C. K. Yap. Complete subdivision algorithms, ii: Isotopic meshing of singular algebraic curves. *Journal of Symbolic Computation*, 47(2):131 – 152, 2012.

[6] M. A. Burr. Continuous amortization and extensions: With applications to bisection-based root isolation. *Journal of Symbolic Computation*, 77:78 – 126, 2016.

[7] M. A. Burr and F. Krahmer. Sqfreeeval: An (almost) optimal real-root isolation algorithm. *Journal of Symbolic Computation*, 47(2):153 – 166, 2012.

[8] M. A. Burr, F. Krahmer, and C. K. Yap. Continuous amortization: A non-probabilistic adaptive analysis technique. Technical Report TR09-136, Electronic Colloquium on Computational Complexity, 2009.

[9] J.-S. Cheng, X.-S. Gao, and L. Guo. Root isolation of zero-dimensional polynomial systems with linear univariate representation. *Journal of Symbolic Computation*, 47(7):843–858, 2012.

[10] J.-S. Cheng, K. Jin, and D. Lazard. Certified rational parametric approximation of real algebraic space curves with local generic position method. *Journal of Symbolic Computation*, 58:18–40, 2013.

[11] S.-W. Cheng, T. Dey, E. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *SIAM journal on computing*, 37(4):1199–1227, 2007.

[12] G. E. Collins and A. G. Akritas. Polynomial real root isolation using Descarte's rule of signs. In *Proc. of the 3rd ACM Int'l Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 272–275, 1976.

[13] C. D'Andrea, T. Krick, and M. Sombra. Heights of varieties in multiprojective spaces and arithmetic nullstellensätze. *Annales scientifiques de l'École Normale Supérieure*, 46(4):549–627, 2013.

[14] J. Dedieu and J. Yakoubsohn. Computing the real roots of a polynomial by the exclusion algorithm. *Numerical Algorithms*, 4(1):1–24, 1993.

[15] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the descartes method. In *Proc. of the Int'l Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 71–78, 2006.

[16] G. Elber and M.-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *Proc. 6th ACM Symposium on Solid Modeling and Applications*, pages 1–10. ACM, 2001.

[17] M. Giusti, G. Lecerf, B. Salvy, and J.-C. Yakoubsohn. On location and approximation of clusters of zeros of analytic functions. *Foundations of Computational Mathematics*, 5(3):257–311, 2005.

[18] W. Heiden, T. Goetze, and J. Brickmann. Fast generation of molecular surfaces from 3d data fields with an enhanced "marching cube" algorithm. *Journal of Computational Chemistry*, 14(2):246–250, 1993.

[19] P. Henrici. Methods of search for solving polynomial equations. *Journal of the Association for Computing Machinery*, 17(2):273–283, 1970.

[20] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.

[21] L. Lin, C. Yap, and J. Yu. Non-local isotopic approximation of nonsingular surfaces. *Computer-Aided Design*, 45(2):451–462, 2013.

[22] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[23] A. Mantzaflaris, B. Mourrain, and E. P. Tsigaridas. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. *Theoretical Comput. Sci.*, 412(22):2312–2330, 2011.

[24] B. Mourrain and J. Pavone. Subdivision methods for solving polynomial equations. *Journal of Symbolic Computation*, 44(3):292–306, 2009.

[25] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 245–254, 2004.

[26] H. Ratschek and J. Rokne. *Computer methods for the range of functions*. Ellis Horwood Series: Mathematics and its Applications. Halsted Press, 1984.

[27] M. Sagraloff and C. Yap. A simple but exact and efficient algorithm for complex root isolation and its complexity analysis. In *Proc. of the 36th Int'l Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 353–360, 2011.

[28] P. Schröder. Subdivision as a fundamental building block of digital geometry processing algorithms. In *In 15th Toyota Conference: Scientific and Engineering Computations for the 21st Century - Methodologies and Applications*, 2002.

[29] V. Sharma, G. Vegter, and C. Yap. Isotopic arrangement of simple curves: an exact numerical approach based on subdivision. http://www.cs.nyu.edu/exact/doc/svy-curve-arrangement.pdf, 2011.

[30] J. M. Snyder. Interval analysis for computer graphics. In *Proc. of the 19th Annual conference on Computer graphics and interactive techniques*, pages 121–130, 1992.

[31] M. Sombra. *Estimaciones para el teorema de ceros de Hilbert*. PhD thesis, Universidad de Buenos Aires, 1998.

[32] J. von zur Gathen and J. Gerhard. Fast algorithms for taylor shifts and certain difference equations. In *Proc. Int'l Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 40–47. ACM, 1997.

[33] C. Wang, Y.-J. Chiang, and C. Yap. On soft predicates in subdivision motion planning. In *Proc. of the 20th Annual Symposium on Computational Geometry*, pages 349–358, 2013.

[34] C. Wang, Y.-J. Chiang, and C. Yap. On soft predicates in subdivision motion planning. *Computational Geometry*, 48(8):589–605, 2015.

[35] J. Yakoubsohn. Approximating the zeros of analytic functions by the exclusion algorithm. *Numerical Algorithms*, 6(1):63–88, 1994.

[36] J.-C. Yakoubsohn. Numerical analysis of a bisection-exclusion method to find zeros of univeriate analytic functions. *Journal of Complexity*, 21(5):652–690, 2005.

[37] C. Yap and L. Lin. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete & Computational Geometry*, 45(4):760–795, 2011.

[38] C. Yap, M. Sagraloff, and V. Sharma. Analytic root clustering: A complete algorithm using soft zero tests. In *Conference on Computability in Europe*, pages 434–444. Springer, 2013.

[39] C. Yap, V. Sharma, and J.-M. Lien. Towards exact numerical Voronoi diagrams. In *Proc. 9th Int'lon . Voronoi Diagrams in Science and Engineering (ISVD)*, pages 2–16, 2012.