A Geometrical Root Finding Method for Polynomials, with
Complexity Analysis

Juan-Luis García Zapata, Departamento de Matemáticas
Juan Carlos Díaz Martín, Departamento de Tecnología de los
Computadores y las comunicaciones
Escuela Politécnica, Universidad de Extremadura
10003, Cáceres (Spain)

Abstract

The usual methods for root finding of polynomials are based on
the iteration of a numerical formula for improvement of successive
estimations. The unpredictable nature of the iterations prevents to
search roots inside a pre-specified region of complex plane. This work
describes a root finding method that overcomes this disadvantage. It
is based on the winding number of plane curves, a geometric con-
struct that is related to the number of zeros of each polynomial. The
method can be restrained to search inside a pre-specified region. It
can be easily parallelized. Its correctness is formally proven, and its
computational complexity is also analyzed.

Keywords: Root Finding, Winding Number, Quadtree Algorithm, Con-
tour Integration.
1 Introduction: Root Finding as a Geometric Problem

In this section we first compare the iterative methods of root finding with the geometric methods, then we display a graphic example of the disadvantages of the former ones, and finally expose the practical motivation of the geometric method shown in this paper.

The methods to find polynomial roots can be roughly classified as iterative or as geometric. The iterative methods (IM) are based on a sequence of error estimation and correction, which, in well-defined conditions, leads to a complex point as close to a root as needed. The methods of this type are fast (better than linear convergence, in general) and its analysis (that is, the proof of its correction, and the determination of the resources needed) relies on relatively standard numerical techniques ([23]). They are the common approach to the polynomial root finding problem, named after Newton, Müller, Legendre, etc. For example, the procedure built in mathematical packages (as LAPACK [1] or MATLAB [27]) is an IM which is based on the eigenvectors of the companion matrix of the polynomial (see [6]). An IM usually converges rapidly for most polynomials of moderate degree (less than about 50 [21]). However, even for these degrees, the IM are inadequate for certain classes of polynomials (as those with clusters or with multiple roots, depending on the method), or for specific polynomials that show ill-conditioning (like the Wilkinson example [30]). To cope with these issues, it is necessary to apply heuristic rules in case of difficulties (to choose another initial approximations, or change the method applied, or use high precision
arithmetic [21]). These matters hinder the algorithmic complexity analysis of IM. For example, for the Newton method there is not such complexity analysis (see [23], [5]). However, in practice most applications of root finding rely on IM, with an proper set of heuristics.

Another disadvantage of IM is that the position of the found root is not related to the initial estimation [3]. This can be pictured in a figure of basins of attraction. The \textit{basin of attraction} of a root (for the Newton method) is the set of complex values \( z \) such that the method converges to that root, if we start with initial value \( z \). It is said that a root attracts the points that converge to it. The basins of two distinct roots are disjoint subsets of the plane. We depict the basins of the roots of a 31-degree polynomial in figure [1]. It is the plane region \(|\text{Re}(z)| < 1.2, |\text{Im}(z)| < 1.2\). The roots are marked with stars. The Newton method is applied one time for every pixel of the figure. The initial root estimation is the complex value corresponding to the position of this pixel. Each pixel is colored with a gray level, according to the root to which it converges. Hence each basin is colored with a different gray level.

In the figure, the boundaries between basins show points whose color is different from that of its neighborhoods. One such point converges to a root located far apart from the roots corresponding to nearby points. Then the root to which a point converges is unpredictable. This \textit{non-local convergence} is characteristic of IM. It is a phenomenon of sensitive dependence of initial conditions, and it cannot be avoided with any level of precision [13].

In this setting, to find for instance the roots near the unit circumference with common IM, it is necessary to find all roots, and pick up those of
Figure 1: Basins of attraction for the Newton method applied to a 31 degree polynomial. The stars mark the roots, and the gray levels its basins.

interest. To avoid the repeated finding of the same root, each found root is removed from the polynomial by dividing it by a linear factor (a so called deflation step). The deflation steps require high precision arithmetic, which is computationally intensive \cite{23}.

In contrast with IM, others methods of root finding are based on geometric notions about the distribution of roots in the complex plane, as its relative position, or if they are contained inside a given circle, etc. Tradition-
ally, geometric methods (GM) are applied in a first stage of root finding, as “rules of thumb” to separate intervals on the real line (or to separate regions on the complex plane) each containing a desired root. On a second stage, an iterative method (that now in each separated interval supposedly converges) approximates the root in every interval. Tools used for root separation in the real line include the criterion of Cauchy for root signs, the bisection method (based on Bolzano’s theorem), or the Sturm successions \[23\]. Examples of root separation tools in the complex plane are Lehmer-Schur \[17\], Weyl \[11, 31\], or the process of Graeffe \[2\]. The GM are more difficult to implement than iterative methods, requiring data types for geometric objects and complex variables, and tree-search or backtracking procedures for control flow. Notwithstanding, the methods based on geometrical relations are valid for all polynomials equally. This uniformity allows an analysis of the complexity of such methods. The theoretical studies of complexity have been a driving force in the development of GM \([24]\). Suppose that we require the roots of a polynomial with up to \(b\) bits, that is, with a precision of \(2^{-b}\). The number of multiplications needed to extract all the roots of a \(n\)-degree polynomial, with this precision, is \(O(n^2 \log n \log b)\) using the GM of \([21]\), or \(O(n \log n \log b)\) in \([18]\).

Contrarily to what happens with IM, the use of deflation steps (and hence of high precision) is not necessary in GM. Besides, if the area of interest is relatively reduced, and contains few roots, a GM avoids spending computations in not desired roots. Many practical applications need methods that allow to focus the root finding on a specified complex plane region (see \([26]\), \([29]\) and references therein). This ability produces a computational saving
with respect to IM.

In our work [8] we face a situation where a geometric method is better than an iterative one. This advantage mainly comes from that in many cases there is not need to find all and each root of a polynomial, but only those ones satisfying certain conditions. In signal processing applications, a polynomial represents an autoregressive model (Linear Predictive Coding, or LPC) of the signal [19]. The roots of this polynomial are inside the unit circle, by the stability of the model, and the roots that are closer to the circumference are related to the main frequency components of the signal. For instance the polynomial of figure 1 comes from LPC of a voice signal. We are interested in finding the complex roots of LPC polynomials that are placed near the unit circumference, up to a distance that depends on the degree of the modeling. This leads us to consider GM for this problem. In signal processing there are several IM to find the roots of greater module, as Graeffe or Bernoulli [23]. However, the high degree of the polynomials of interest for us, and the intricate case analysis required by these methods, are two features that obstruct its application, and the practical methods of voice processing must use spectral analysis techniques.

In general, the GM have a common pattern. Let us consider, for expository purpose, the bisection method for zero finding of real continuous functions. It is based on the Bolzano’s theorem. This theorem says that if a continuous function \( f : [a, b] \to \mathbb{R} \) in a interval \([a, b]\) of the real straight takes opposed signs at its ends (that is, \( f(a) \cdot f(b) < 0 \)), then it has at least a root in the interval. The bisection method consists of using this fact recursively to build a succession of nested intervals (and hence progressively smaller)
containing a real root. These are the prototype components of a GM: an inclusion test, to decide whether there is any root in a piece (either of the line or of the plane); and a recursive procedure to get progressively smaller localizations of desired roots.

We discuss two examples for comparison: Lehmer-Schur and Weyl methods. As the GM are based on the delimitation of regions that can contain roots, they are often called bracketing methods, by analogy with the bisection ones. The Lehmer-Schur method \cite{17} is a bi-dimensional bracketing method based on an inclusion test by Schur and Cohn, which says whether a polynomial has roots inside a circle. The area of interest is covered by circles, and this criterion is applied to each one. Then, the circles that contain a root are covered by circles of lesser radius, and so on, until the required precision is reached. The recursive procedure entails certain inefficiency because it implies overlapping and repeating finding of the same roots. The bracketing method of Weyl \cite{11} is based on the decomposition in squares of the domain of interest. Each square is subdivided in four sub-squares, if it contains some root, and so recursively. This leads to a quadtree procedure, a search tree in which each node is a square that is linked with its four sub-squares. Weyl and others \cite{21} proposed several inclusion tests to see if there are roots inside a square. So, “method of Weyl” is a generic name of several GM, with a common recursive procedure (the quadtree), but with different inclusion tests.

The theoretically optimal methods (in the sense of the computational complexity of the root finding problem) are of geometric type \cite{24, 20, 18, 25}, but they are not widespread in practice. The implementation of
GM is usually enriched with some iterative stages, like Newton when the quadtree search reaches a subregion with fast convergence. Other changes in practical implementations of a GM blueprint are related with numerical issues of the inclusion tests. In the implementations in floating-point arithmetic, that is, with a fixed finite precision, the roundoff errors can accumulate \cite{9}. This can make that a test miss a root if, for example, the test indicate the presence of a root with a zero value and its absence with a non-zero value. However, if we switch to an implementation in arbitrary-precision arithmetic, with no roundoff errors, other drawback arises: the precision (and hence the computational resources needed) can grow indefinitely \cite{21}. This problems of numerical accuracy have been an obstacle to the practical use of GM.

In this work we expose a method with a floating point implementation that avoids the precision problem noted above. It is based on an inclusion test, IPSR in figure \cite{9} that is applicable to regions of arbitrary shapes, and on a recursive procedure giving a quadtree search, RDP in figure \cite{10}. The inclusion test uses the winding number of closed plane curves. This number is the amount of twists of the curve around the origin. We have developed \cite{7} an efficient procedure to compute the winding number, and we use it in IPSR. Besides, we give a bound of the computational cost of the test. In the recursive procedure RDP, we avoid the situations that require the use of IPSR with high precision, and hence also avoid the precision problem. We compute a bound of the cost of the full root finding method using the cost bound of the inclusion test ISPN in each node of the RDP quadtree search.

About the structure of the paper, section 2 below exposes the procedure of winding number computation. We give results about its applicability and
cost, mainly in terms of the distance to the origin from the curve to which the procedure applies. In section 3 we adapt these results to the curves that appear in the root finding method. The cost bound in this case is more complex, in terms of the distance to every root from the curve. The section 4 describe the quadtree procedure, RDP. The recursive partition ought to be performed avoiding to cross over polynomial roots, because this cause precision problem in IPSR. The section 5 contains a cost analysis of the full root finding method.

2 Winding Number as inclusion test

We compute the winding number of a curve using a discrete sampling obtained by an iterative procedure. We have proved the convergence of this procedure for nonsingular curves in [7], and we also modified it in such a way that the singular curves will not cause a endless computation.

Let us consider closed curves defined parametrically, that is, as mappings of an interval to the complex plane \( \Delta : [a, b] \to \mathbb{C} \) with \( \Delta(a) = \Delta(b) \). The winding number, or index, \( \text{Ind}(\Delta) \) of a curve \( \Delta : [a, b] \to \mathbb{C} \), is the number of complete rotations, in counterclockwise sense, of the curve around the point \((0, 0)\). See figure 2

As a particular case of the Cauchy formula of Complex Analysis, the winding number is equal to this line integral:

\[
\text{Ind}(\Delta) = \frac{1}{2\pi i} \int_{\Delta} \frac{1}{w} dw
\]

The principle of the argument [11] states that the number of zeros (counted
The winding numbers of the curves $\Delta_1$ and $\Delta_2$ are $\text{Ind}(\Delta_1) = 1$ and $\text{Ind}(\Delta_2) = 2$.

with multiplicity) of an analytic function $f : \mathbb{C} \to \mathbb{C}$, $w = f(z)$, inside a region with border defined by the curve $\Gamma$, is equal to the winding number of the curve $\Delta = f(\Gamma)$. The principle of the argument can be viewed as a bi-dimensional analogy of Bolzano’s theorem, and it is in the base of several recursive methods to find the zeros of holomorphic functions [10].

It should be noted that the winding number of the curve $\Delta$ is not defined if $\Delta$ crosses over the origin $(0, 0)$. In that case $\Delta$ is called singular curve, since the integral $\int_{\Delta} \frac{1}{w} \, dw$ does not exist. If $\Delta = f(\Gamma)$, this is equivalent to that $\Gamma$ crosses over a root of $f$. In that case $\Gamma$ is called singular curve with respect to $f$, the integral $\int_{\Delta} \frac{1}{w} \, dw$ does not exist, and the above theorem is not applicable.

We apply the principle of the argument to analytic functions defined by polynomials. The number of zeroes $N_0$ of a polynomial complex function $f(z)$ inside a region bounded by a curve $\Gamma$ is the winding number of the curve $f(\Gamma)$, that is: $N_0 = \text{Ind}(f(\Gamma))$. For an example, see figure 3. This can be used as an inclusion test to decide if a given region of the plane has any
root, that is, $N_0 > 0$.

Figure 3: The number of roots of the polynomial $f(z) = z^3 + 1$ inside $\Gamma_1$ and $\Gamma_2$ equals the winding numbers of $\Delta_1 = f(\Gamma_1)$ and $\Delta_2 = f(\Gamma_2)$.

The computation of the winding number is not an easy task. We could say that there are two schools of thought: the approaches that use the numerical integration, as [16], and those based on computational geometry ([11]). The first school use the integration error estimates to ensure the correct calculation of the winding number. The second school, while producing faster algorithms, need to assume some hypothesis to prove their correctness. We follow this school in the method of [7], and proof its correction under a mild hypothesis of Lipschitzianity. We now summarize the method and give a bound of its cost.

A curve $\Delta$ is defined by a continuous mapping $\Delta : [a,b] \to \mathbb{C}$. We also call $\Delta$ to the set of its points, so $x \in \Delta$ means that there is $t \in [a,b]$ with $x = \Delta(t)$. Numerically, it is represented by a polygonal approximation, that is, a discrete set of complex points, followed in certain order. Figure 4 shows
the curve $\Delta$ with two polygonal approximations.

Figure 4: The curve $\Delta$ is approximated by polygonal $\tilde{\Delta}_9$ with a resolution of 9 points, and by $\tilde{\Delta}_{18}$ with double resolution.

To calculate the winding number $\text{Ind}(\Delta)$ of a contour $\Delta$, we divide the complex plane in angular sectors, for example of angle $\pi/4$, like in figure 5. There are eight such sectors, called $C_0, C_1, C_2, \ldots , C_7$, each sector the half of a quadrant. A division in angular sectors with angle different of $\pi/4$ is also possible. The following method and the proof of its correction are similar in this case, provided that there are as least four sectors, and only the constants involved must be changed.
Let us suppose that the curve \( \Delta \) has a defined index, that is, it does not cross the origin. Then it is sure that there is an array \((t_0, t_1, \ldots, t_m)\) of values of the parameter \( t \in [a, b] \), \( a = t_0 < t_1 < \cdots < t_m = b \) whose images by the mapping \( \Delta \) are in adjacent sectors.

We say that a sector is *connected* to other sector if they are adjacent or are the same. Two points are *connected* if they are placed in connected sectors. A polygon of vertexes \( \Delta(t_i), i = 0, 1, \ldots, m \) satisfies the property of connection if any two consecutive points \( \Delta(t_i) \) and \( \Delta(t_{i+1}) \) are connected.
That is, the polygon verifies that if $\Delta(t_i) \in C_x$ and $\Delta(t_{i+1}) \in C_y$ then $y = x \pm 1$ (or $y = x$). As the sectors $C_7$ and $C_0$ are adjacent, the equalities must be understood as an arithmetic congruence modulo 8.

Henrici [11] suggests that if we obtain by some method an array of points $(\Delta(t_0), \Delta(t_1), \ldots, \Delta(t_m))$ defining a polygon $\widetilde{\Delta}_m$ that verifies the property of connection, then its index $\text{Ind}(\widetilde{\Delta}_m)$ can be calculated as the number of points $\Delta(t_i)$ in $C_7$ followed by a point $\Delta(t_{i+1})$ in $C_0$. The occurrence of a sector $C_0$ followed by $C_7$ must be counted negatively, standing for a turning back in the curve. That is, $\text{Ind}(\widetilde{\Delta}_m) = \#(\text{crossings } C_7 \text{ to } C_0) - \#(\text{crossings } C_0 \text{ to } C_7)$.

The procedure to find the parameter values $(t_0, t_1, \ldots, t_m)$ was left unspecified by Henrici, as well as the conditions under which the index of the approximating polygon $\widetilde{\Delta}_m$, defined by the $m$ points array $(\Delta(t_0), \Delta(t_1), \ldots, \Delta(t_m))$, coincides with that of the original curve $\Delta$.

The paper [32] of Ying and Katz devises a procedure to find an array $(\Delta(t_0), \Delta(t_1), \ldots, \Delta(t_m))$ verifying the property of connection, at a reasonable computational cost. It constructs such array starting from some initial samples $(a = s_0, \ldots, s_n = b)$ of the curve $\Delta$, whose images do not necessarily verify the property of connection. This means that perhaps, for some $i$, the images of $s_i$ and $s_{i+1}$ are not placed in connected sectors. The array of the images of $(s_0, \ldots, s_n)$ is scanned from the beginning $s_0$, until a pair $(s_i, s_{i+1})$ is found such that its images $\Delta(s_i) \in C_x$ and $\Delta(s_{i+1}) \in C_y$ are in non-connected sectors. In this situation an interpolation point $\frac{s_i + s_{i+1}}{2}$ is inserted in the array $(s_0, \ldots, s_n)$ between $s_i$ and $s_{i+1}$. Then the resulting array $(s'_0, \ldots, s'_{n+1})$ is scanned again for another pair $(s'_j, s'_{j+1})$ whose images are not in connected sectors and a middle point is inserted as described. Iterating
this process, finally an array \((t_0, \ldots, t_m), \ m \geq n\), is obtained such that the polygon of its images \((\Delta(t_0), \ldots, \Delta(t_m))\) verifies the property of connection.

This basic procedure of Ying-Katz has two features that prevent its practical application. First, it can run forever if it is applied to singular curves. Besides, in nonsingular curves, it can produce a result that is not correct, because a curve segment \(\Delta(s_i, s_{i+1})\) that surrounds the origin can have its endpoints connected, hence is undetected by the procedure. For example, in figure 6 \(\Delta\) and \(\tilde{\Delta}\) have different winding number, by a lost turn between \(s_i\) and \(s_{i+1}\).

In [7] we proposed a procedure that enriches the loop of scanning-insertion of the basic procedure of Ying-Katz with an additional entry condition, that ensures the correct calculation of the winding number by the net number of \(C7-C0\) crossings of the result array. Being precise, for a curve \(\Delta : [a, b] \to \mathbb{C}\) Lipschitz of constant \(L\), and an array \(S = (s_0, s_1, \ldots, s_n)\), the assertion \(p(s_i)\) is “the values \(s_i\) and \(s_{i+1}\) in array \(S\) have their images \(\Delta(s_i)\) and \(\Delta(s_{i+1})\) not connected” and the assertion \(q(s_i)\) is “the values \(s_i\) and \(s_{i+1}\) in array \(S\) verify \((s_{i+1} - s_i) \geq |\Delta(s_i)| + |\Delta(s_{i+1})|/L\)”. The meaning of \(s_{i+1}\) in the extreme case of \(i = n\) is \(s_0\). With this notation, we have the Insertion Procedure (so called IP) of figure 7.

Informally, we can say that the insertion procedure involves a while loop that is repeated until a connected array (i.e., verifying the condition “no \(p\)”) and without lost turns (“no \(q\)”) is obtained.

The validity of the output array to compute \(\text{Ind}(\Delta)\) is proved in [7]. In this work the cost of IP is also analyzed, using the notion of \(\varepsilon\)-singularity. For a value \(\varepsilon \geq 0\), we say that a curve is \(\varepsilon\)-singular if its minimum distance
Theorem 1. If $\Delta : [a, b] \to \mathbb{C}$ is $\varepsilon$-singular with $\varepsilon \neq 0$, with a Lipschitzian parameterization of constant $L$, then the IP for the curve $\Delta$ with initial array $S$ concludes in less than 

\[ 4L(b-a) \left\lceil \frac{L|S|}{\varepsilon} \right\rceil \left\lceil \log_2 \left( \frac{4L|S|}{\pi \varepsilon} \right) \right\rceil + \left( \frac{4L|S|}{\pi \varepsilon} + 1 \right) \left\lceil \frac{L(b-a)}{\varepsilon} \right\rceil \] 

iterations, and the returned array gives us the wind-
**Insertion procedure:** To find the winding number of a curve
\( \Delta : [a, b] \to \mathbb{C} \)

**Input parameters:** The curve \( \Delta \) of Lipschitz constant \( L \), and
an array \( S = (a = s_0, s_1, \ldots, s_n = b) \), sampling of \([a, b]\).

**Output:** An array that is valid to compute \( \text{Ind}(\Delta) \).

**Method:**

While there is a \( s_i \) in \( S \) with \( p(s_i) \) or with \( q(s_i) \) do:

\[
\text{Insert } \frac{s_i + s_{i+1}}{2} \text{ between } s_i \text{ and } s_{i+1};
\]

Return the resulting array.

**Figure 7:** Insertion procedure (IP).

Pushing aside the constants, this cost bound is of order
\[
O \left( \frac{1}{\varepsilon^2} \log \left( \frac{1}{\varepsilon} \right) + \frac{1}{\varepsilon^2} \right) = O \left( \frac{1}{\varepsilon^2} \log \left( \frac{1}{\varepsilon} \right) \right). \]

However, this result is of little interest for the practical
application of IP, because the value \( \varepsilon \) is not known a priori.

The above theorem says us that the number of iterations is bounded by a
dercreasing function of \( \varepsilon \). Reciprocally, we can deduce that if the number of
iterations surpasses certain threshold, the value \( \varepsilon \) should be low, that is, the
curve should be near the origin. Following this viewpoint, we define another
procedure that extends IP with a check to detect a small value of \( \varepsilon \) and
exits in that case (that corresponds to a curve close to singular). It is called
insertion procedure with control of singularity (IPS). IPS introduces the new
assertion \( r(s_i, Q) \) that depends on a input parameter \( Q \), as shown in figure 8.
$r(s_i, Q)$ is defined as “the values $s_i$ and $s_{i+1}$ in array $S$ verify $s_{i+1} - s_i \leq Q$”. The parameter $Q$ controls the maximum number of iterations that IPS can reach.

**Insertion procedure with control of singularity:** To find the winding number of a curve $\Delta : [a, b] \to \mathbb{C}$

**Input parameters:** The curve $\Delta$ with Lipschitz constant $L$, an array $S = (s_0, s_1, \ldots, s_n)$, sampling of $[a, b]$, and a real parameter $Q > 0$.

**Output:** An array that is valid to compute $\text{Ind}(\Delta)$, on normal exit, or a value $t$ in $[a, b]$ such that $|\Delta(t)| \leq \frac{LQ}{\sin(\pi/8)}$, if exit on error.

**Method:**

While there is a $s_i$ in $S$ with $p(s_i)$ or with $q(s_i)$ do:

{ Insert $\frac{s_i + s_{i+1}}{2}$ between $s_i$ and $s_{i+1}$;  
  If $r(s_i, Q)$, return either $t = s_i$ or $t = s_{i+1}$ depending on $\min(|\Delta(s^{(k)}_i)|, |\Delta(s^{(k)}_{i+1})|)$. [Exit on error]  
}

Return the resulting array. [Normal exit]

Figure 8: Insertion procedure with control of singularity (IPS).

In [7] is also proved that:

**Theorem 2.** If $\Delta : [a, b] \to \mathbb{C}$ is Lipschitzian with constant $L$, $S$ an array sampling of $[a, b]$, and $Q$ a positive real, the insertion procedure with control of singularity applied to $\Delta$, $S$ and $Q$ verifies:

a) It returns in less than $\left\lfloor \frac{b - a}{Q} \right\rfloor$ iterations.
b) If it exits normally then the returned array gives us $\text{Ind}(\Delta)$.

c) If it exits on error, the returned value $t$ verifies $|\Delta(t)| \leq \frac{LQ}{\sin\left(\frac{\pi}{8}\right)}$.

Theorem 2 sets that IPS effectively computes the index of curves $\Delta$ with $\varepsilon > \frac{LQ}{\sin\left(\frac{\pi}{8}\right)}$. For curves whose distance to the origin is under this level, IPS can by chance return normally with an array valid to compute the index, or it can return with error, signaling that the input curve is $\varepsilon$-singular. In any case, the procedure returns in less than $\left\lfloor \frac{b-a}{Q} \right\rfloor$ iterations.

3 Insertion procedure for curves $\Delta = f(\Gamma)$

The computational cost of a winding number computation using IP (that is, the number of iterations) is related to the distance of the curve from the origin $d(O, \Delta)$ (theorem 1). The IPS, in addition, when the cost exceeds a threshold, returns with error with a bound on this distance (theorem 2).

In this section we focus on curves of the form $\Delta = f(\Gamma)$. These curves arise when we compute the number of roots of $f$ contained inside $\Gamma$. In this particular case, the cost of applying IP to $\Delta$ given by theorem 1 can be put in function of the distance of the curve $\Gamma$ from the nearest root of $f$. We will show that the bound of this distance that we can obtain from theorem 2 is not adequate for our purpose. In consequence, we define another procedure (IPSR, theorem 3) that, when it returns on error, gives us a bound of certain function of the roots and the curve (the condition number). This bound will be useful in the next section, for the iterative method of partition of the search region.
To compute the winding number of curves $\Delta = f(\Gamma)$ using IP, the theorem requires that $\Delta$ be Lipschitzian. To ensure that this requirement is fulfilled, we impose that $\Gamma$ be uniformly parameterized. This is not a restriction in practice because the curve $\Gamma$ that encloses the region of interest is usually built connecting straight segments (or circumference arcs) uniformly parameterized. Also, if this plane region is compact, its boundary $\Gamma$ is bounded and, consequently, by a result about differentiable functions in a bounded set ([15]), $f$ is Lipschitzian (on the set $\Gamma$) with constant $L = \sup_{x \in \Gamma} |f'(\Gamma(x))|$. Hence $\Delta = f(\Gamma)$ is the composition of an uniformly parameterized curve and a Lipschitzian function, and so $\Delta$ is a Lipschitzian curve with the same constant $L$. Therefore the hypothesis of theorem 1 is verified, and IP computes the winding number of $\Delta = f(\Gamma)$ (i.e., the number of roots inside $\Gamma$).

The key factor in the cost of the winding number computation, by theorem 1, is the distance to the origin from $\Delta$. This distance and the roots of $f$ are related as described in the following proposition.

Let $f(z) = a_n z^n + \cdots + a_1 z + a_0$ be a polynomial of degree $n$ and its root decomposition $f(z) = a_n (z - z_1)(z - z_2)\ldots(z - z_n)$. Recall that the distance $d(A, B)$ between two sets $A$ and $B$ is the minimum of the distances between each pair of points of the respective sets. In particular, if $Z$ is the set of the roots of $f$, $Z = \{z_1, z_2, \ldots, z_n\}$, its distance $d(Z, \Gamma)$ to $\Gamma$ is the distance from the closer root, and $d(O, \Delta)$ is the distance from the origin $O$ to $\Delta$.

**Proposition 1.** If $\Gamma : [a, b] \to \mathbb{C}$ is a curve uniformly parameterized, $f$ a polynomial of degree $n$, with Lipschitz constant $L$, and $\Delta = f(\Gamma)$, then:

$$|a_n| d(Z, \Gamma)^n \leq d(O, \Delta) \leq L d(Z, \Gamma)$$
Remember that if we apply IP to compute the number of roots inside $\Gamma$, its cost will be $O\left(\frac{1}{\varepsilon^2} \log \left(\frac{1}{\varepsilon}\right)\right)$ by the bound of Theorem 1, where $\varepsilon = d(O, \Delta)$.

In terms involving $\Gamma$, by the inequality $\frac{1}{|a_n| d(Z, \Gamma)^n} \geq \frac{1}{d(O, \Delta)}$ deduced from the above proposition, the computation has a cost of order lesser or equal than

$$O\left(\frac{1}{|a_n| d(Z, \Gamma)^n} \log \left(\frac{1}{|a_n| d(Z, \Gamma)^n}\right)\right)$$

that is $O\left(\frac{1}{d(Z, \Gamma)^{2n}} \log \left(\frac{1}{d(Z, \Gamma)^n}\right)\right)$.

As the distance $d(Z, \Gamma)$ to the roots is unknown, we confront an undetermined cost using IP. This was the motivation to introduce IPS (figure 8), which with we have an assured bound of cost of $\left\lfloor \frac{(b - a)}{Q} \right\rfloor$ iterations, by theorem 2 since $\Delta = f(\Gamma)$ and $\Gamma$ is defined over $[a, b]$. In addition to this assured cost, a second advantage of IPS is that if the bound of cost is reached, it returns on error with a point of the curve near to origin, $|\Delta(t)| \leq \frac{LQ}{\sin \left(\frac{\pi}{8}\right)}$, as theorem 2 says. This gives us a bound to $d(O, \Delta)$, that is $\frac{LQ}{\sin \left(\frac{\pi}{8}\right)}$. In the particular case of curves of the form $\Delta = f(\Gamma)$, we can deduce a bound to $d(Z, \Gamma)$: by the proposition 1, $|a_n| d(Z, \Gamma)^n \leq d(O, \Delta) \leq |\Delta(t)| \leq \frac{LQ}{\sin \left(\frac{\pi}{8}\right)}$, and hence $d(Z, \Gamma) \leq \sqrt[n]{\frac{LQ}{|a_n| \sin(\pi/8)}}$, in case of error exit of IPS.

In case of return with error we need a bound finer than this on $d(Z, \Gamma)$, to locate a root close to the point that produce the error. Unfortunately, the bound given by IPS is very loose, because it decreases much more slowly than the parameter $Q$: its value can be near to 1 for moderate values of $n$ even with a very small $Q$. Taking the $n$-th root is a severe handicap in the formula of a bound. The procedure IPS, as least to the extent that is covered
by theorem \(2\) does not provide this bound at a reasonable cost.

We must define another procedure for which a bound exists that depends on \(Q\) without \(n\)-th root extraction. In the Insertion Procedure with control of Singularity for number of Roots (IPSR, figure 9), we use the predicates \(p\), \(q_2\) and \(r\) meaning \(p(s_i)\) that “the values \(s_i\) and \(s_{i+1}\) in array \(S\) have their images \(f(\Gamma(s_i))\) and \(f(\Gamma(s_{i+1}))\) not connected”, \(q_2(s_i)\) is “the values \(s_i\) and \(s_{i+1}\) in array \(S\) verify:

\[
|f(\Gamma(s_i))| + |f(\Gamma(s_{i+1}))| \leq 2 |f'(\Gamma(s_i))| (s_{i+1} - s_i) + |f(\Gamma(s_{i+1})) - f(\Gamma(s_i))|,
\]

and \(r(s_i, Q)\) means “the values \(s_i\) and \(s_{i+1}\) in array \(S\) verify \((s_{i+1} - s_i) \leq Q\)”. As before, \(s_{i+1}\) in the extreme case of \(i = n\) should be intended as \(s_0\). The assertions \(p\) and \(r\) are equal than in IPS, but \(q_2\) is different from \(q\). The assertion \(q_2\) is required in proposition \(2\) to prove that there are not lost turns and in proposition ?? to show that certain bound in error return is verified.

The return of the IPSR verifies a claim that involves \(\kappa_f(\Gamma)\), the condition number of the curve \(\Gamma\) with respect to \(f\), defined as the sum of inverses of the distances from \(\Gamma\) to each root of \(f\):

\[
\kappa_f(\Gamma) = \sum_{i=1}^{n} \frac{1}{d(z_i, \Gamma)}.
\]

The role of condition number \(\kappa_f(\Gamma)\) in IPSR is similar to that of the inverse of \(d(O, \Delta)\), the singularity of \(\Delta\), in IPS. By theorem \(2\) an error return of IPS implies a low value of \(d(O, \Delta)\), that is, a high value of \(\frac{1}{d(O, \Delta)}\). We will show that an error return of IPSR implies a high value of \(\kappa_f(\Gamma) = \sum_{i=1}^{n} \frac{1}{d(z_i, \Gamma)}\).

With the following proposition we will prove (in theorem \(3\)) that if IPSR exit normally, the returned array gives the number of roots inside \(\Gamma\).
**Insertion procedure with control of singularity for the number of roots:** To find the number of roots of a polynomial $f$ inside a curve $\Gamma : [a, b] \to \mathbb{C}$

**Input parameters:** The curve $\Gamma$ uniformly parameterized, the polynomial $f$ of degree $n$ and Lipschitz constant $L$, an array $S = (s_0, \ldots, s_n)$, sampling of $[a, b]$, and a real parameter $Q > 0$.

**Output:** An array $(t_0, \ldots, t_m)$ that is valid to compute $\text{Ind}(f(\Gamma))$ on normal exit. A value $t \in [a, b]$ such that $\kappa_f(\Gamma)$ is greater than $\frac{\sin(\pi/8)}{Q}$, if exit on error.

**Method:**

While there is a $s_i$ in $S$ with $p(s_i)$ or with $q_2(s_i)$ do:

{ Insert $\frac{s_i + s_{i+1}}{2}$ between $s_i$ and $s_{i+1}$;  
  If $r(s_i, Q)$, return. [Exit on error]
}

Return the resulting array. [Normal exit]

Figure 9: Insertion procedure with control of singularity for number of roots (IPSR).
Proposition 2. For a curve \( f(\Gamma) \), if \( q_2(s_i) \) in array \( S = (\ldots, s_i, s_{i+1}, \ldots) \) is not verified, then there is not a lost turn between \( s_i \) and \( s_{i+1} \).

We now classify the iterations performed by the “while” loop. An iteration is of type \( p \) if it is performed because the property \( p(s_i) \) is verified, and it is of type \( q \) if it is performed because the property “not \( p(s_i) \), and \( q_N(s_i) \)” is verified. So any iteration is of type \( p \) or of type \( q \), but not both. Likewise the error exits can be classified as of type \( p \), or \( q \), according to the type of the iteration in which the return occurs.

We will show in theorem 3 that IPSR exits with an array valid to compute the number of roots inside \( \Gamma \), or with a lower bound of the condition number (that is, the analogous of theorem 2 for IPS). Previously we give bounds on this condition on errors of type \( p \) and \( q \) in two propositions after several auxiliary lemmas. We consider the angles measured in the interval \([0, 2\pi)\).

Lemma 1. If \( \alpha \) is the oriented angle between three complex numbers \((x, z, y)\) with vertex at \( z \) (that is, \( \alpha = \arg(z - y) - \arg(z - x) \)) then

\[
\min(d(z, x), d(z, y)) \leq \frac{d(x, y)}{2|\sin(\alpha/2)|}
\]

The accuracy and cost of the IPSR is described by the following theorem:

Theorem 3. If \( \Gamma : [a, b] \to \mathbb{C} \) is uniformly parameterized, \( S \) an array sampling of \([a, b] \), \( Q \) a positive real, and \( f \) a polynomial of degree \( n \) with Lipschitz constant \( L \) in \( \Gamma \), the insertion procedure with control of singularity, IPSR, verifies:

a) Returns in less than \( \left\lfloor \frac{b - a}{Q} + 1 \right\rfloor \) iterations.
b) If it exits normally then the returning array gives the number of roots of \( f \) inside \( \Gamma \).

c) If it exits on error, then \( \kappa_f(\Gamma) \geq \frac{\sqrt{2}}{4Q} \).

The condition number \( \kappa_f(\Gamma) \) and the distance of \( \Gamma \) from the nearest root of \( f \) are related in the following way:

**Proposition 3.** If \( \kappa_f(\Gamma) \geq \frac{\sqrt{2}}{4Q} \) then there is a root \( z_i \) with \( d(z_i, \Gamma) \leq \frac{4nQ}{\sqrt{2}} \).

The insertion procedure with control of root proximity prevents an excessive number of iterations, controlled by input parameter \( Q \). The IPSR effectively computes the number of roots inside \( \Gamma \) if there are not close roots. But if there are roots on \( \Gamma \) or near it, the procedure can return with error, signaling this fact, or can return normally with an array valid to compute the number of roots, always in less than \( \left\lfloor \frac{b - a}{Q} + 1 \right\rfloor \) iterations.

Perhaps it is also possible to proof that it returns normally in less than \( O(\kappa_f(\Gamma)^2 \log(\kappa_f(\Gamma))) \) iterations, a result similar to theorem 1. It should be noted that this dependence on the distance to the roots is consistent with other algorithms that compute the number of roots in a region \([24, 21]\). In any case, this use of the condition number \( \kappa_f(\Gamma) \) for theoretical analysis is not necessary for our purposes of root finding.

### 4 Recursive use of the procedure

In this section we detail a geometric algorithm for root finding, following the common pattern depicted in the introduction, that is, an inclusion test to decide if there is a root in a region, and a recursive procedure (RDP, figure
to subdivide the region and locate the roots with the needed accuracy. With the result of theorem 3, the IPSR computes the number of roots inside \( \Gamma \) by the winding number of \( \Delta = f(\Gamma) \). This procedure will be used as inclusion test. We now describe the recursive procedure.

The size of a region \( P \) is measured by its rectangular diameter \( \text{dm}_{\text{Rect}}(P) \), that is precisely defined in subsection 4.1. An approximation of a root up to an accuracy of \( A > 0 \) is a region of rectangular diameter lesser than \( A \) that contains the root.

The Recursive Division Procedure (RDP, figure 10) divides the initial region into subregions progressively smaller, until the diameter goes below the accuracy. These regions are inserted into the array \( R \), which at the end of RDP contains approximations to the roots. Each region \( P_i \) included in \( R \) is labeled with the number of roots that it contains, \( n_i \). The case of \( n_i > 1 \) corresponds to either a multiple root or a cluster (a set of roots with distance lesser than \( A \) between them).

Two items of the RDP in figure 10 must be precised. The first item is the computation of \( \text{windN}(P) \), the winding number of the image by \( f \) of the border of \( P \). This computation is made by \( \text{IPSR}(P,Q) \) for some value of \( Q \), as previously mentioned. This value \( Q \) must be chosen as a compromise between a large value (that gives us lower computational cost by theorem 3a), and a value small enough to not cause an error (Theorem 3b) in the initial region or in the subregions that arise in the recursive decomposition. The computation must be done without error because the winding number is needed for processing a region (that is, either discarding or decomposing it).
**Recursive Division Procedure:** To find the roots of a polynomial \( f \) inside a region \( P \).

**Input parameters:** A region \( P \) of the complex plane, a polynomial \( f \), and a parameter of accuracy \( A > 0 \).

**Output:** The array \( R = (P_1, P_2, \ldots, P_k) \) (containing \( k \) approximations up to an accuracy of \( A > 0 \) of all the roots of \( f \) in \( P \)) and the array \( N = (n_1, n_2, \ldots, n_k) \) (of numbers \( n_i \geq 1 \) such that each \( P_i \) contains \( n_i \) roots of \( f \) counting multiplicity).

**Global structures used:** Two arrays \( R \) and \( N \), initially empty.

**Method:** \( \text{RDP}(P, A) \) {
    If \( \text{windN}(P) = 0 \), return. [exit 1]
    If \( \text{dm}_{\text{Rect}}(P) < A \) then
        Add \( P \) to \( R \) and \( \text{windN}(P) \) to \( N \), and return. [exit 2]
    Else
        Divide \( P \) in four subregions \( P_1, P_2, P_3 \) and \( P_4 \).
        For each \( i = 1, 2, 3, 4 \), do
            \( \text{RDP}(P_i, A) \).
        [exit 3]
    }

Figure 10: Recursive Division Procedure (RDP) for polynomial root finding. Two items remain to be specified: the parameter \( Q \) to compute \( \text{windN}(P) \) using \( \text{IPSR}(P, Q) \), and the decomposition method of “Divide P”.

27
The second item is the decomposition “Divide P”, which makes four subregions $P_1, P_2, P_3$ and $P_4$ from $P$. This should be done with the following three requirements: a) the subregions obtained $P_1, P_2, P_3$ and $P_4$ must be a set partition of $P$ (that is, $P = \bigcup P_i$ and $P_i \cap P_j = \emptyset$), so at the end each root will be in a component of $R$ and only in one, b) the diameter must be decreasing, $d_{Rect}(P_i) < d_{Rect}(P)$, and c) each region obtained $P_i$ must be such that IPSR($P_i, Q$) returns without error for the chosen value $Q$, as discussed above. We will describe a decomposition method and a value of $Q$ such that these requirements are fulfilled.

In the next subsection 4.1 we make a first try of a decomposition method “Divide P”. We cut successively in half the region, first horizontally and then vertically, and we will show that the resulting subregions are decreasing in rectangular diameter. However the arising subregions can cause an error in IPSR. In the following subsection 4.2 we make a second and definitive try of method “Divide P”, and show that a value of $Q$ lesser than $\frac{A \sin(\pi/8)}{n_0 n 4 \sqrt{2}}$ does not produce error with IPSR in the subregions, being $n_0$ the number of roots in the original region. In the final subsection 4.3 we show that the RPD ends in a finite number of calls, that the claim on the output of the figure 10 is verified at the end, and we calculate its computational cost.

Before this, a comment about the type of regions that we consider. For simplicity, we require that the initial region, as well as those that arise in its subdivisions, are plane regions whose border is a simple closed curve $\Gamma$ (Jordan curve [15]). These regions are connected. A non connected region is, for instance, the interior of two disjoint circles, and its border (two circumferences) is not a a simple curve. Non connected regions can be treated.
in a way similar to the connected ones, because the number of roots inside a non connected region is the sum of the winding numbers of its connected components. However, the assumption of connectedness simplify the reasonings.

We will produce the subregions by cuts along straight lines. To prevent such cuts to cause a subregion non connected, we enforce that the initial region $P$ is convex (i.e., that if a straight segment has its endpoints in $P$, all the segment is inside $P$). So, the subregions will be convex (hence connected). To find the roots in a non convex region, the RDP should be applied to its convex hull, or to its decomposition into convex parts.

### 4.1 Divide into Smaller Pieces: a first try

We consider each plane region as a closed set, that is, including its border. For a plane region $P$, its top horizontal supporting line $l_T$ is the upper horizontal straight line that has some point in common with $P$. Likewise, the bottom horizontal supporting line $l_B$ of $P$ is the lower horizontal straight line that has some point in common with $P$. The vertical diameter $d_{\text{V}}(P)$ is the distance between these lines $d(l_T, l_B)$. That is $d_{\text{V}}(P) = \min_{x \in l_T, y \in l_B} d(x, y)$.

In a similar manner, the left vertical supporting line $l_L$ is the leftmost vertical straight line that has some point in common with $P$, and the right vertical supporting line $l_R$ is the rightmost straight line that has some point in common with $P$. The horizontal diameter $d_{\text{H}}(P)$ is the distance between these lines $d(l_L, l_R)$. That is $d_{\text{H}}(P) = \min_{x \in l_L, y \in l_R} d(x, y)$. The figure [11] depicts these concepts, and also the classical diameter of $P$ defined by $d_{\text{m}}(P) = \max_{x, y \in P} d(x, y)$.
For regions defined by straight segments, as the polygonals arising in the IPSR, the horizontal and vertical diameters are easier to compute than the classical diameter. They are related by the following bounds.

**Lemma 2.** If two plane regions $P_1, P_2$ verify $P_1 \subset P_2$, then $\text{dm}(P_1) \leq \text{dm}(P_2)$, $\text{dm}_H(P_1) \leq \text{dm}_H(P_2)$, and $\text{dm}_V(P_1) \leq \text{dm}_V(P_2)$.

**Proof.** For the classical diameter is clear because the maximum, in the definition of $\text{dm}(P_2)$, is taken over a greater set than in $\text{dm}(P_1)$. For the horizontal diameter, note that the two vertical supporting lines of $P_1$ are between the two vertical supporting lines of $P_2$, perhaps coinciding with some of them.
Hence its distance is lesser. Likewise for the vertical diameter.

\[ \square \]

We also define the \textit{rectangular diameter} \( d_{\text{Rect}}(P) = \sqrt{d_{H}(P)^2 + d_{V}(P)^2} \).

**Proposition 4.** For a plane region \( P \), it is verified:

\[
\max(d_{H}(P), d_{V}(P)) \leq d(P) \leq d_{\text{Rect}}(P)
\]

**Proof.** For the first inequality, note that the supporting lines of \( P \) have at least one point in common with \( P \). Being \( x_0 \) one of these points for the upper horizontal supporting line, that is \( x_0 \in P \cap l_T \), and \( y_0 \) for the lower one, \( y_0 \in P \cap l_B \), using the definitions of \( d_{V}(P) \) and \( d(P) \) we have \( d_{V}(P) = \min_{x \in l_T, y \in l_B} d(x, y) \leq d(x_0, y_0) \leq \max_{x, y \in P} d(x, y) = d(P) \). Likewise \( d_{H}(P) \leq d(P) \), hence \( \max(d_{H}(P), d_{V}(P)) \leq d(P) \).

For the second inequality, we define the \textit{HV-envelope} of \( P \), \( \text{Env}_{HV}(P) \), as the bounded rectangle delimited by the horizontal and vertical supporting lines. As \( P \subset \text{Env}_{HV}(P) \), by the lemma 2 we have that \( d(P) \leq d(\text{Env}_{HV}(P)) \). Furthermore, the diameter of the rectangle \( \text{Env}_{HV}(P) \), of base \( d_{H}(P) \) and height \( d_{V}(P) \), is the distance between opposed vertexes, then \( d(\text{Env}_{HV}(P)) = \sqrt{d_{H}(P)^2 + d_{V}(P)^2} = d_{\text{Rect}}(P) \). Chaining with the previous inequality, we conclude. \( \square \)

To divide a figure in lesser parts, we define the operators \( T, B, L \) and \( R \) acting on a plane region \( P \). If \( m_{H}(P) \) is the straight line in the middle between the horizontal supporting lines, \( T(P) \) is the intersection of \( P \) with the upper half-plane defined by \( m_{H}(P) \), and \( B(P) \) is the intersection of \( P \) with the lower half-plane. Likewise, If \( m_{V}(P) \) is the straight line in the middle
between the vertical supporting lines, $L(P)$ is the intersection of $P$ with the
left half-plane defined by $m_V(P)$, and $R(P)$ is the intersection of $P$ with the
right half-plane. The operators $T$ and $B$ are said of type \textit{horizontal}, while $L$
and $R$ are said of type \textit{vertical}. The picture \ref{fig:linalg:operators} shows several compositions
of these operators applied to a non convex region.

![Diagram showing the operators applied to a region]

Figure 12: Applying the operators $T$, $B$, $L$, $R$ to $P$.

\textbf{Lemma 3.} The operators decrease the diameters, verifying:

\[
\begin{align*}
\text{dm}_H(T(P)) & \leq \text{dm}_H(P), & \text{dm}_H(L(P)) & \leq \frac{\text{dm}_H(P)}{2}, \\
\text{dm}_V(T(P)) & \leq \frac{\text{dm}_V(P)}{2} & \text{and} & \text{dm}_V(L(P)) \leq \text{dm}_V(P)
\end{align*}
\]

The same formulas are valid changing $T$ to $B$ and $L$ to $R$.

\textit{Proof.} For the horizontal diameter, the first inequality is a consequence of
lemma \ref{lem:linalg:basic-proposition}. The second comes from that the vertical supporting lines of $L(P)$
are between \( l_L \) and \( m_V(P) \), hence the horizontal diameter are lesser or equal than the distance \( d(l_L, m_V(P)) \), that is the half of \( dm_H(P) \).

For the vertical diameter, the reasoning is similar, and also for operators \( B \) and \( R \).

\( \square \)

We will find the rate of diameter decreasing under the alternating applications of horizontal and vertical divisions to the initial region. An operator is of type \( O_m \), for \( m \geq 1 \), if it is the composition of \( m \) operators of type horizontal alternating with \( m \) of type vertical, starting with an horizontal operator. That is, \( O \) is of type \( O_m \) if \( O = V_m H_m V_{m-1} H_{m-1} \cdots V_1 H_1 \), being \( H_i \in \{T,B\} \) and \( V_i \in \{L,R\} \). There are \( 2^{2m} \) operators of type \( O_m \). (See figure 13).

**Lemma 4.** For \( m \geq 1 \), if \( O_m \) is an operator of type \( O_m \)

\[
\text{dm}_{Rect}(O_m(P)) \leq \frac{\text{dm}_{Rect}(P)}{2^m}
\]

**Proof.** By induction on \( m \). We denote as above with \( H \) an operator that can be \( T \) or \( B \), and with \( V \) another that can be \( L \) or \( R \). For \( m = 1 \), we have that \( O = VH \). Chaining some inequalities of lemma 3 several times:

\[
\text{dm}_{Rect}(VH(P)) = \sqrt{\text{dm}_H(VH(P))^2 + \text{dm}_V(VH(P))^2} \leq \\
\leq \frac{\sqrt{\text{dm}_H(H(P))^2}}{2} + \sqrt{\text{dm}_V(H(P))^2} \leq \frac{\sqrt{\text{dm}_H(P)^2 + \text{dm}_V(P)^2}}{2} = \frac{\text{dm}_{Rect}(P)}{2}
\]

For \( m > 1 \), note that if \( O_m \) is of type \( O_m \), then \( O = VH O_{m-1} \) with \( O_{m-1} \) of type \( O_{m-1} \). Applying again proposition 4 and lemma 3 several times, we
Figure 13: The regions arising from applying all the operators in $O_2$ to a region.

have:

$$dm_{Rect}(O_m(P)) =$$

$$= dm_{Rect}(VHO_{m-1}(P)) \leq \sqrt{dm_H(VHO_{m-1}(P))^2 + dm_V(VHO_{m-1}(P))^2} \leq$$

$$\leq \sqrt{\frac{dm_H(HO_{m-1}(P))^2}{2^2} + dm_V(HO_{m-1}(P))^2} \leq$$

$$\leq \sqrt{\frac{dm_H(O_{m-1}(P))^2}{2^2} + \frac{dm_V(O_{m-1}(P))^2}{2^2}} \leq$$

$$\leq \sqrt{\frac{dm_H(O_{m-1}(P))^2 + dm_V(O_{m-1}(P))^2}{2}}$$
This, applying the hypothesis of induction, is lesser or equal than:

\[
\sqrt{\left(\frac{\text{dm}_H(P)}{2^{m-1}}\right)^2 + \left(\frac{\text{dm}_V(P)}{2^{m-1}}\right)^2} = \frac{\sqrt{\text{dm}_H(P)^2 + \text{dm}_V(P)^2}}{2^m} = \frac{\text{dm}_{\text{Rect}}(P)}{2^m}
\]

In particular, \(\text{dm}_{\text{Rect}}(O_{m+1}(P)) \leq \frac{\text{dm}_{\text{Rect}}(O_m(P))}{2}\). With respect to the classical diameter, in general it is not true that \(\text{dm}(O_m(P)) \leq \frac{\text{dm}(P)}{2^m}\). For example if \(P\) is the circle of radius 1, \(\text{dm}(P) = 2\) but \(TR(P)\) is a circular sector with \(\text{dm}(TR(P)) = \sqrt{2} > \frac{\text{dm}(P)}{2}\). However we have the following:

**Corollary 1.** If \(O_m\) is an operator of type \(O_m\)

\[\text{dm}(O_m(P)) \leq \frac{\text{dm}_{\text{Rect}}(O_m(P))}{2}\]

**Proof.** Applying proposition \(\Box\) \(\text{dm}(O_m(P)) \leq \text{dm}_{\text{Rect}}(O_m(P))\), and chaining with the above lemma.

By the above corollary, we have that the division of the region by horizontal and vertical cuts effectively reduces the diameter of the obtained parts.

In the context of the recursive procedure RDP of figure \(\Box\) we define the decomposition in subregions “Divide \(P\)” as \(P_1 = RT(P), P_2 = LT(P), P_3 = RB(P)\) and \(P_4 = LB(P)\). That is, we first cut horizontally, and then \(T(P)\) and \(B(P)\) are cut vertically. By lemma \(\Box\) for \(m = 1\), the obtained regions will satisfy the requirement that \(\text{dm}_{\text{Rect}}(P_i) < \text{dm}_{\text{Rect}}(P)\). Besides, the lemma \(\Box\) for \(m > 1\) also describes the size of regions of later recursive calls.
4.2 Divide into Smaller Pieces: second try

The decomposition method described above produces subregions decreasing in rectangular diameter. However, if we apply IPSR to these subregions, it can return on error without finishing the computation, as theorem 3 established. The following is an example of this situation.

Let us consider a circumference $\Gamma$ centered slightly at the right of the origin, that contains the roots of the polynomial $f(z) = z^3 - 1$. Its image $\Delta = f(\Gamma)$ encircles three times the origin (see figure 14). The origin is marked with $\circ$, and it is the image of the three roots. We can suppose for the sake of the example that the IPSR, with a reasonable parameter $Q$, computes the winding number of $\Delta$, that is 3. Following RDP in figure 10, we decompose the interior of $\Gamma$ in four circular sectors, by an horizontal and a vertical cut. One of these subregions has a border crossing over a root, like the depicted in figure 14 $c$). The image of the border of this subregion, say $f(\Gamma_s)$, crosses over the origin (figure 14 $d$), and hence the IPSR can not compute its winding number.

The setting of figure 14 $c$), with a root exactly in the border $\Gamma_s$, can be unlikely, but a an exit with error can also arise if the subregion border has a condition number $\kappa_f(\Gamma_s)$ high enough, by theorem 3 $c$). This implies, by proposition 3, that there is a root near $\Gamma_s$. In short, if the IPSR applied to $f(\Gamma_s)$ with parameter $Q$ returns with error, there is a root at less that $\frac{nQ}{\sin(\pi/8)}$ from $\Gamma_s$, and it does not perform the winding number computation.

Therefore, the division into four subregions proposed previously is incorrect. To divide in a way that no error is returned by IPSR, we proceed as
follows. Suppose that we start with $\text{RDP}(P,A)$, from an initial region $P$, that does not produce error with IPSR. To divide $P$ in subregions also without error for IPSR, let us consider first the subregions that arise from $P$ by an horizontal cut following $m_H(P)$, that is $T(P)$ and $B(P)$. If one of these
regions produces an error in IPSR, then the condition number of its border is greater than $\frac{\sin(\pi/8)}{Q}$ by theorem 3. and there is a point in the border at less than $\frac{nQ}{\sin(\pi/8)}$ from a root by proposition 3. This point belongs to $m_H(P)$, because the rest of the border of $T(P)$ or $B(P)$ is a border of $P$ too, and this region did not produce an error.

In such case, with $m_H(P)$ near a root, say $z_1$, we do not divide $P$ using $m_H(P)$, but another line instead that does not give rise to an error exit of IPSR. It is sure that the horizontal line that is at $\frac{2nQ}{\sin(\pi/8)}$ above $m_H(P)$ is at distance greater than $\frac{nQ}{\sin(\pi/8)}$ from $z_1$. Perhaps this second line causes again an error in the IPSR, because there are another root $z_2$ near it. In such case the division is performed following a third horizontal line below $m_H(P)$, at distance $\frac{2nQ}{\sin(\pi/8)}$ (figure 15).

This process, moving the horizontal line a distance of $\frac{2nQ}{\sin(\pi/8)}$ to avoid the roots, will be performed as much $n_0$ times, being $n_0$ the number of roots inside $P$. The half of the $n_0$ or less trial lines are above $m_H(P)$ and the other half below it. As they are $\frac{2nQ}{\sin(\pi/8)}$ apart, the final horizontal cutting line is at less than $n_0 \cdot \frac{nQ}{\sin(\pi/8)}$ from $m_H(P)$, and does not give place to an IPSR error. A similar reasoning is valid for vertical cutting lines. The division with shifted lines of $P$ is done first with an horizontal cut without error, an then each of the two obtained parts is cut vertically without error.

We have described the division of the initial region $P$, supposing that IPSR($P,Q$) does not return with error. This means that there are no roots near the border of the region of interest, which is a reasonable assumption for $P$. In any case, IPSR must be applied to $P$ before RPD to ensure that it
Figure 15: Being $E = \frac{n}{\sin(\pi/8)}$, the lines $m_H(P)$, $m_H(P) + 2EQ$, $m_H(P) - 2EQ$, $m_H(P) + 4EQ$ successively cause error by proximity of the roots $z_1$, $z_2$, $z_3$ and $z_4$ respectively. Finally $m_H(P) - 4EQ$ does not have a root at distance lesser than $\frac{nQ}{\sin(\pi/8)}$.

does not fail.

This method of division with shifted lines is recursively applied to the subregions. In the initial region, the assertion that it does not produce IPSR error must be ensured previously, as commented, but for the subregions the equivalent assertion is obtained as a consequence of the cuts shifting.

Remember that the lemma 4 gives us a reduction in diameter for divisions along $m_H(P)$ and $m_V(P)$. The division by the other shifted lines also produces a decreasing in the diameter, excluding an additive term. We develop a general result about diameters of iterated shifted cuts (in lemma 7). Applying this lemma to our way of avoiding the roots (figure 15) we can give
a value of \( Q \) (in proposition 5) such that the subdivisions are decreasing and besides the calls to IPSR in this subregions do not return with error.

We define the operator \( T \) in the following way. The horizontal line at distance \( \lambda \) above \( m_H(P) \) is denoted \( m_{H+\lambda}(P) \). \( T_\lambda(P) \) is the intersection of \( P \) with the upper half-plane defined by \( m_{H+\lambda}(P) \). If \( \lambda \) is a negative value, the line \( m_{H+\lambda}(P) \) should be understood below \( m_H(P) \). Analogously the operator \( B_\lambda(P) \) is the intersection of \( P \) with the lower half-plane defined by the same horizontal line, \( m_{H+\lambda}(P) \). In this way \( T_\lambda(P) \cup B_\lambda(P) = P \) and \( T_\lambda(P) \cap B_\lambda(P) \) is its common border, a segment of \( m_{H+\lambda}(P) \). Likewise \( R_\lambda(P) \) is the intersection of \( P \) with the right half-plane defined by \( m_{V+\lambda}(P) \) (the vertical line at distance \( \lambda \) at right of \( m_V(P) \)), and \( L_\lambda(P) \) the intersection with the left half-plane defined by the same line \( m_{V+\lambda}(P) \). See figure 16.

**Lemma 5.** If \( \lambda \geq 0 \), then \( B(P) \subset B_\lambda(P) \) and \( R(P) \subset R_\lambda(P) \). If \( \lambda \leq 0 \), then \( B(P) \supset B_\lambda(P) \) and \( R(P) \supset R_\lambda(P) \). For the operators \( T_\lambda \) and \( L_\lambda \), in the same hypothesis the inclusions are reversed.

**Proof.** It is immediate considering the position of \( m_{H+\lambda}(P) \) and \( m_H(P) \) (or of \( m_{V+\lambda}(P) \) and \( m_V(P) \)).

\( \square \)

We say that the operators \( T_\lambda \) and \( B_\lambda \) are of type *horizontal* and \( L_\lambda \) and \( R_\lambda \) of type *vertical*. Let us denote with \( H_\lambda \) a generic horizontal operator, that is, either \( T_\lambda \) or \( B_\lambda \). Similarly \( V_\lambda \) denotes either \( L_\lambda \) or \( R_\lambda \). Note that for \( \lambda \) large enough, the result of an operator can be the empty set, whose diameter is conventionally zero.
Lemma 6. For an horizontal operator $H_\lambda$, $dm_H(H_\lambda(P)) \leq dm_H(P)$. If besides $|\lambda| \leq \frac{dm_V(P)}{2}$ then $dm_V(H_\lambda(P)) \leq \frac{dm_V(P)}{2} + |\lambda|$.

For an vertical operator $V_\lambda$, $dm_V(V_\lambda(P)) \leq dm_V(P)$. If besides $|\lambda| \leq \frac{dm_H(P)}{2}$ then $dm_H(V_\lambda(P)) \leq \frac{dm_H(P)}{2} + |\lambda|$.

Proof. The first inequality is a consequence of lemma 2. The condition $|\lambda| \leq \frac{dm_V(P)}{2}$ makes that the defining line of $H_\lambda(P)$ will be placed at less than $|\lambda|$ from $m_H(P)$, above if either $\lambda \geq 0$ and $H_\lambda = T_\lambda$, or $\lambda \leq 0$ and $H_\lambda = B_\lambda$, and below in other cases. Anyway the second inequality is verified. Similarly the claim about vertical operators.

Figure 16: The action of the operators $T_\lambda$, $B_\lambda$, $L_\lambda'$ and $R_\lambda'$.
An operator $O$ is of type $O_{m,\mu}$ for $m \geq 1$ and $\mu \geq 0$ if it is the composition of $m$ operators $H_{\lambda_i}$ and $m$ $V_{\lambda'_i}$, $i = 1, \ldots, m$, alternating, with $|\lambda_i| \leq \mu, |\lambda'_i| \leq \mu$, starting with an horizontal operator. That is $O = V_{\lambda_m} H_{\lambda_m} V_{\lambda_{m-1}} H_{\lambda_{m-1}} \cdots V_{\lambda_1} H_{\lambda_1}$. The next lemma shows that the diameters of the regions $O(P)$ are lesser than that of $P$, except for an additive term.

**Lemma 7.** For $m \geq 1, \mu \geq 0$, if $O_m$ is an operator of type $O_{m,\mu}$ then

$$dm_{\text{Rect}}(O_m(P)) \leq \frac{dm_{\text{Rect}}(P)}{2^m} + \frac{2^m - 1}{2^{m-1}} \mu \sqrt{2}$$

**Proof.** First, note that $dm_{\text{Rect}}(P) = \|(d_H(P), d_V(P))\|$, being $\|\|$ the Euclidean plane norm. Also, by the triangle inequality $\|(a + \mu, b + \mu)\| \leq \|(a, b)\| + \|(\mu, \mu)\| = \|(a, b)\| + \mu \sqrt{2}$.

We prove the lemma by induction on $m$. For $m = 1$, we have that $O = V_{\lambda_1} H_{\lambda_1}$. Hence, applying some inequalities of lemmas 3 and 6,

$$dm_{\text{Rect}}(V_{\lambda_1} H_{\lambda_1}(P)) = \sqrt{dm_H(V_{\lambda_1} H_{\lambda_1}(P))^2 + dm_V(V_{\lambda_1} H_{\lambda_1}(P))^2} \leq \sqrt{\left(\frac{dm_H(H_{\lambda_1}(P))}{2} + |\lambda_1|\right)^2 + dm_V(H_{\lambda_1}(P))^2} \leq \sqrt{\left(\frac{dm_H(P)}{2} + |\lambda_1|\right)^2 + \left(\frac{dm_V(P)}{2} + |\lambda_1|\right)^2} \leq \sqrt{\left(\frac{dm_H(P)}{2} + \mu\right)^2 + \left(\frac{dm_V(P)}{2} + \mu\right)^2} = \left\|\left(\frac{dm_H(P)}{2} + \mu, \frac{dm_V(P)}{2} + \mu\right)\right\| \leq \frac{dm_{\text{Rect}}(P)}{2} + \mu \sqrt{2}$$

For $m > 1$, if $O_m$ is of type $O_{m,\mu}$, then $O = V_{\lambda_m} H_{\lambda_m} O_{m-1}$ with $O_{m-1}$ of type $O_{m-1,\mu}$. In the first case (the second are similar), applying again the lemmas and the hypothesis of induction, we have:

$$dm_{\text{Rect}}(O_m(P)) = dm_{\text{Rect}}(V_{\lambda_m} H_{\lambda_m} O_{m-1}(P)) \leq \frac{dm_{\text{Rect}}(O_{m-1}(P))}{2} + \mu \sqrt{2}$$
By the hypothesis of induction, $\text{dm}_{\text{Rect}}(O_{m-1}(P)) \leq \frac{\text{dm}_{\text{Rect}}(P)}{2^{m-1}} + \frac{2^{m-1} - 1}{2^{m-2}} \mu \sqrt{2}$, hence

$$\frac{\text{dm}_{\text{Rect}}(O_{m-1}(P))}{2} + \mu \sqrt{2} \leq \frac{\text{dm}_{\text{Rect}}(P)}{2^{m-1}} + \frac{2^{m-1} - 1}{2^{m-2}} \mu \sqrt{2} + \mu \sqrt{2} = \frac{\text{dm}_{\text{Rect}}(P)}{2^m} + \frac{2^{m-1} - 1}{2^{m-1}} \mu \sqrt{2} + \mu \sqrt{2} = \frac{\text{dm}_{\text{Rect}}(P)}{2^m} + \frac{2^m - 1}{2^m} \mu \sqrt{2}$$

Note that $\frac{2^m - 1}{2^{m-1}} = \left(2 - \frac{1}{2^{m-1}}\right) < 2$, hence $\text{dm}_{\text{Rect}}(O_m(P)) < \frac{\text{dm}_{\text{Rect}}(P)}{2^m} + 2 \sqrt{2} \mu$.

For any values $\lambda, \lambda_1'$ and $\lambda_2'$ lesser or equal than $\mu$, we say that the subregions $P_1 = R_{\lambda_1'}T_{\lambda}(P), P_2 = L_{\lambda_1'}T_{\lambda}(P), P_3 = R_{\lambda_2'}B_{\lambda}(P)$ and $P_4 = L_{\lambda_2'}B_{\lambda}(P)$ are a $\mu$-decomposition of $P$. That is, we first cut horizontally with a $\lambda$ shift, and then $T_{\lambda}(P)$ and $B_{\lambda}(P)$ are cut vertically with respective shifts $\lambda_1'$ and $\lambda_2'$. The separation of the cut lines $m_{H+\lambda}(P), m_{V+\lambda}(P)$ from the center lines $m_H(P), m_V(P)$ has a maximum value of $\mu$.

The last term of the inequality of lemma \[8\] $2 \sqrt{2} \mu$, obstructs the decreasing of the rectangular diameters of the regions that arise after successive $\mu$-decompositions. After any number of $\mu$-decompositions, the rectangular diameter can be greater than this last term.

We apply this theory of $\mu$-decompositions to our approach to avoiding the roots (figure \[15\]), and show how to get a decreasing in rectangular diameter up the desired precision even considering the obstruction of lemma \[7\]. Remember that we want to reach a subregion of diameter lesser than $A$, with cuts shifted a maximum separation of $\mu \leq \frac{n_0 n Q}{\sin(\pi/8)}$, being $n_0$ the number of roots inside the original region $P$. We have that:
Proposition 5. If \( Q \leq \frac{A \sin(\pi/8)}{4\sqrt{2n_0n}} \), there is a number \( m \) such that any operator \( O_m \) of type \( O_{m,\mu} \) verifies \( \text{dm}_{\text{Rect}}(O_m(P)) \leq A \).

Proof. If we apply to \( P \) an operator \( O_m \) of type \( O_{m,\mu} \), by lemma 7

\[
\text{dm}_{\text{Rect}}(O_m(P)) \leq \frac{\text{dm}_{\text{Rect}}(P)}{2^m} + 2\mu\sqrt{2}
\]

First, note that for some \( m \), \( \frac{\text{dm}_{\text{Rect}}(P)}{2^m} \leq \frac{A}{2} \). Second, with \( Q \leq \frac{A \sin(\pi/8)}{4\sqrt{2n_0n}} \), as \( \mu \leq \frac{n_0nQ}{\sin(\pi/8)} \), we have

\[
2\sqrt{2}\mu \leq 2\sqrt{2}\frac{n_0nA\sin(\pi/8)}{4\sqrt{2n_0n}} = \frac{A}{2}
\]

Adding these bounds:

\[
\frac{\text{dm}_{\text{Rect}}(P)}{2^m} + 2\mu\sqrt{2} \leq \frac{A}{2} + \frac{A}{2} = A
\]

as we wanted to show.

\[\square\]

4.3 Termination and cost of the recursive procedure

The following theorem assures that the recursive procedure RPD of figure 10 ends and that its output verifies the claimed property. To make reference to the nested calls of RPD, we say that the first call has recursion level 0, and the calls to the same function made inside a call of recursion level \( v \) has recursion level \( v + 1 \).

Theorem 4. If \( P \) is a plane region containing \( n_0 \) roots of the polynomial \( f \), with \( \text{dm}_{\text{Rect}}(P) > A \) and such that IPSN\( (P, Q) \) does not produce error, the procedure RPD applied to it with an accuracy of \( A \) verifies that:

a) It ends, reaching a level of recursion lesser or equal than \( \log_2 \left( \frac{\text{dm}_{\text{Rect}}(P)}{A} \right) + 2 \).
b) At the end, the plane regions of array $R = (P_1, P_2, \ldots, P_k)$ are approximations of all the roots of $f$ in $P$, each containing the number of roots given by $N = (n_1, n_2, \ldots, n_k)$.

Proof. For $a)$, by proposition $5$, choosing $Q \leq \frac{A \sin(\pi/8)}{4\sqrt{2n_0n}}$ we have that $2\sqrt{2}\mu \leq \frac{A}{2}$ and, by lemma $7$, $\frac{\text{dm}_{\text{Rect}}(O_m(P))}{2^m} \leq \frac{A}{2}$ for any operator $O_m$ of type $O_{m,\mu}$. Note that in the recursion level $v$ the method of decomposition apply operators of type $O_{m,\mu}$ with $m = v$. Then, to decrease the rectangular diameter under $A$ is sufficient to reach a recursion level $v$ verifying $\frac{\text{dm}_{\text{Rect}}(P)}{2^v} + \frac{A}{2} \leq A$, that is $\frac{\text{dm}_{\text{Rect}}(P)}{2^v} \leq A$.

When the subregion $O_m(P)$ that arise in a call have its diameter below $A$, the subsequent call to RPD returns non recursively by exit 2 (figure 10). Therefore we can reach a recursion level $v$ with $\frac{\text{dm}_{\text{Rect}}(P)}{2^v} \leq \frac{A}{2}$, but the next call will returns by exit 2 (or exit 1 if it is the case) without posterior recursive calls. The maximum recursion level is $m + 1$ with $\frac{\text{dm}_{\text{Rect}}(P)}{2^v} \leq A$, that is

$$\frac{\text{dm}_{\text{Rect}}(P)}{2^v} \leq \frac{A}{2}$$

$$\lg_2(\text{dm}_{\text{Rect}}(P)) \leq v + \lg_2(A) - 1$$

$$\lg_2(\text{dm}_{\text{Rect}}(P)) - \lg_2(A) + 1 \leq v$$

$$\lg_2 \left( \frac{\text{dm}_{\text{Rect}}(P)}{A} \right) + 1 \leq v$$

We have shown that the recursion level reached by RDP is bounded by $\lg_2 \left( \frac{\text{dm}_{\text{Rect}}(P)}{A} \right) + 2$. As the body of the procedure, apart from recursive calls, ends in a finite number of steps, and the recursion level is bounded, the procedure also ends in a finite number of steps.
For b), we prove that at the end of the procedure RDP($P, A$), for any plane region $P$, the arrays $R$ and $N$ verify the assertion “The plane regions of $R$ have diameter lesser than $A$, containing the number of roots given by $N$, and these are all the roots of $f$ contained in $P$”. We will prove this by structural induction ([11, 12]), that is, first in the case of non recursive calls and then, in the other cases, assuming true the corresponding assertions at the end of the four recursive calls RDP($P, A$), i.e. assuming true “The plane regions added to $R$ have diameter lesser than $A$, containing the number of roots given by the number added to $N$, and these are all the roots of $f$ contained in $P$”.

We say that plane region $P$ is of level $v$ if the procedure RDP($P, A$) reach up to recursion level $v$. Every plane region has a finite level, lesser or equal than $\lg_2 \left( \frac{\dim_{\text{Rect}}(P)}{A} \right) + 2$ as we see in a). The structural induction over regions can be viewed as a complete induction (the usual type of induction) over the level $v$ of the regions.

The base case ($v = 0$) are the regions that does not require recursive calls, those without roots (that return by exit 1) or with one or more roots but with diameter lesser than $A$, (that return by exit 2). On returning by exit 1 the assertion is trivially verified because $P$ does not have roots and $R$ (and $N$) remains voids. On returning by exit 2 the assertion is also verified because $R$ contains only the region $P$, and as the IPSR computes without error the winding number, $N$ contain the number of roots inside $P$.

For the general case ($v > 1$) four recursive calls RDP($P_i, A$) are performed, being $P_1, P_2, P_3$ and $P_4$ disjoint regions that cover $P$, of level $v - 1$ or lesser. Lets us call $R_i$ the array of regions added to $R$ by RDP($P_i, A$).
Similarly \( N_i \) is the array of numbers added to \( N \). By induction hypothesis, as \( P_i \) is of level \( v - 1 \), at the end of these calls the regions pertaining to \( R_i \) are all of diameter lesser than \( A \), contains the number of roots given by \( N_i \), and these are all the roots in \( P_i \). Then the concatenation of \( R_i \) and \( N_i \) (that is, \( R \) and \( N \)) verify that its regions has diameter and roots as specified. That these are all the roots inside \( P \) comes from the fact that the \( P_i \) are disjoints and cover \( P \).

\[ \square \]

For the computational cost of the entire root finding method, we count the number of polynomial evaluations that it requires. The determination of \( f(z) \) for a complex \( z \), that is, the polynomial evaluation (PE) is the main operation in the IPSR, in the insertion of a parameter value \( s_i \) in array \( S \), that requires the computation of the point \( f(\Gamma(s_i)) \) to evaluate the properties \( p \) and \( q \) (see figure 9). The rest of the RDP procedure consists of computations of \( m_H(O(P)) \) or \( m_V(O(P)) \) for operators \( O \), in last instance reducible to the mean of maximum and minimum, that is, the traverse of the array of inserted points plus a division. So we consider the cost of these operations negligible compared with the polynomial evaluation of the point themselves.

In general, a polynomial of degree \( n \) can be evaluated in a complex point with \( \left\lfloor \frac{n + 1}{2} \right\rfloor \) multiplications using a complex Horner scheme [14]. Each of these complex multiplications can be performed with three float operations; hence a PE is roughly equivalent to \( \frac{3(n + 1)}{2} \) floats operations.

**Theorem 5.** The number of PE performed by RDP\((P, A)\), for a polynomial \( f \) of degree \( n \) with \( n_0 \) roots inside \( P \), is lesser or equal than

\[
\frac{8\sqrt{2}n_0n}{A \sin(\pi/8)} + (4 + \sqrt{2})\frac{\sqrt{2}n_0n}{\sin(\pi/8)} \left( \lg_2 \left( \frac{dm_{Rect}(P)}{A} \right) + 2 \right).
\]
Proof. We add the number of PE needed by the application of IPSR in the initial region $P$, with those needed by the subregions arising in the recursive calls up to a maximum level of $\lg_2 \left( \frac{\text{dm}_{\text{Rect}}(P)}{A} \right) + 2$. By proposition 5 the values of $Q$ verifying $Q \leq \frac{A \sin(\pi/8)}{4\sqrt{2n_0n}}$ make RDP ends. By theorem 3 $a)$, the number of iterations of IPSR, that is, of PE, is lesser than $\left\lfloor \frac{b - a}{Q} - 1 \right\rfloor$.

The best $Q$ to use in IPSR is the greater value of $Q$ that assure us the termination, that is $Q_0 = \frac{A \sin(\pi/8)}{4\sqrt{2n_0n}}$. With this value, the number of PE required by IPSR in the initial region is lesser than $\left\lfloor \frac{b - a}{Q} - 1 \right\rfloor \leq \frac{b - a}{Q_0}$, that is, $\frac{4\sqrt{2n_0n}(b - a)}{A \sin(\pi/8)}$.

To see how many PE requires the division of a region in four parts, that is, each cut by an horizontal and vertical line, let us consider first the region $T_\lambda(P)$. Calling $\Delta$ to the border of $P$, the number of PE required by $\Delta$ has been considered in the above paragraph. For the border $\Delta_T$ of $T_\lambda(P)$, it is build concatenating a curve segment of $\Delta$ (its upper half) and a straight segment of the cutting line $m_H(H)$. We parameterize uniformly this straight segment, that has length lesser than $\text{dm}_{H}(P)$. The number of PE required by IPSR in $\Delta_T$ come only from the insertion of points in this last segment, because the insertions in the upper half of $\Delta$ were previously made. The number of PE required to insert points at distance lesser than $Q_0$ in a segment of length $\text{dm}_{H}(P)$ is $\frac{\text{dm}_{H}(P)}{Q_0}$, that is equal to $\frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} \text{dm}_{H}(P)$. After the insertion of these points we can apply the IPSR to $T_\lambda(P)$. The region $B_\lambda(P)$ does not require more PE than $T_\lambda(P)$, because its border is composed from the same straight segment (followed in inverse direction) and the lower half of $\Delta$. 

48
Subsequently, the regions $T_{\lambda}(P)$ and $B_{\lambda}(P)$ are themselves divide by vertical cuts. Following the same reasoning, the sum of the number of PE required by the two cuts is lesser than $\frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} (\text{dm}_V(T_{\lambda}(P)) + \text{dm}_V(B_{\lambda}(P)))$, that is lesser or equal than

$$\frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} \left( \frac{\text{dm}_V(P)}{2} + \mu + \frac{\text{dm}_V(P)}{2} + \mu \right)$$

by lemma [3] and using $|\lambda| \leq \mu$. Hence, to construct the four parts (and compute the number of roots inside) when we apply RDP to $P$ demands a number of PE less than $\frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} (\text{dm}_H(P) + \text{dm}_V(P) + 2\mu)$.

After this discussion of the division of the initial region, let us consider the general case. A call to RDP of recursion level $m$ has a entry parameter a region of the form $O_m(P)$ with $O_m \in O_{m,\mu}$, and hence the division in four subregions require $\frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} (\text{dm}_H(O_m(P)) + \text{dm}_V(O_m(P)) + 2\mu)$ PE, similarly to the initial region. By lemma [7] this is lesser or equal than

$$\frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} \left( \frac{\text{dm}_{\text{Rect}}(P)}{2^m} + 2\sqrt{2}\mu + \frac{\text{dm}_{\text{Rect}}(P)}{2^m} + 2\sqrt{2}\mu + 2\mu \right) =$$

$$= \frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} \left( \frac{\text{dm}_{\text{Rect}}(P)}{2^{m-1}} + (4\sqrt{2} + 2)\mu \right) \leq$$

$$\leq \frac{4\sqrt{2n_0n}}{A \sin(\pi/8)} \left( \frac{\text{dm}_{\text{Rect}}(P)}{2^{m-1}} + (1 + \sqrt{2})A \right)$$

The last inequality comes from $\mu \leq \frac{A}{4\sqrt{2}}$.

By theorem [4]a), the maximum recursion level is $l_{\text{max}} = \lg_2 \left( \frac{\text{dm}_{\text{Rect}}(P)}{A} \right) + 2$, and if the initial region $P$ have $n_0$ roots at most $n_0$ branches reach up to this level. Hence adding the above bounds in the number of PE in each level
we have that the total number is lesser or equal than
\[
\sum_{m=1}^{l_{max}} \frac{4\sqrt{2}n_0 n}{A \sin(\pi/8)} \left( \frac{\text{dm}_{\text{Rect}}(P)}{2^{m-1}} + (1 + \frac{\sqrt{2}}{4})A \right) = \\
\frac{4\sqrt{2}n_0 n}{A \sin(\pi/8)} \left( \sum_{m=1}^{l_{max}} \frac{\text{dm}_{\text{Rect}}(P)}{2^{m-1}} + l_{max}(1 + \frac{\sqrt{2}}{4})A \right) < \\
\frac{4\sqrt{2}n_0 n}{A \sin(\pi/8)} \left( 2 \text{dm}_{\text{Rect}}(P) + l_{max}(1 + \frac{\sqrt{2}}{4})A \right) = \\
\frac{8\sqrt{2}n_0 n}{A \sin(\pi/8)} + (4 + \sqrt{2})\frac{\sqrt{2}n_0 n l_{max}}{\sin(\pi/8)}
\]
as we want to show.

\[\square\]

Taking values in the expression, this bound is lesser than
\[
\frac{30n_0 n}{A} + 21n_0 n \left( \lg_2 \left( \frac{\text{dm}_{\text{Rect}}(P)}{A} \right) + 2 \right).
\]

For a given degree \(n\), this bound on the number of PE is of order \(O\left( \frac{1}{A} \right)\).

5 Conclusions

We have develop an algorithm for root finding of polynomials. It allows to restrict the search to an area of interest in the complex plane, unlike the more common iterative algorithms.

The algorithm combines an inclusion test (to see if a plane region contains some roots) based in the winding number, with a recursive method of partition in subregions, like others known algorithms [32, 24, 25, 21, 18]. The contribution is that our method to compute the winding number of curves
[7] detects singular cases (that is, when the curve in question cross over some root). This allows to build sub-regions avoiding these cases, in contrast to other algorithms. Therefore we can also give a formal proof of the correctness of the method.

About the computational complexity, the bound on the number of polynomial evaluations required by the algorithm is of order $O\left(\frac{1}{A}\right)$ with respect the precision $A$. Although our interest in this development is mainly practical, this complexity can be compared with other theoretic bounds, being better than [22] and similar to [18]. In terms of complexity with respect to the degree, which is $O(n_0n)$, is lesser or equal than quadratic, being $O(n \log n)$ that of [18].

In practice the cost of our algorithm can be far lower, because the hypothesis in subsection [4.3] that every root requires maximal level of recursion can be relaxed. Besides, in the geometric root finding methods, is usual to implement a iterative procedure (of faster convergence rate, like Newton’s) to search a root when the diameter of the region are below certain bound which assure this convergence (for example the bound of [28]). This feature is not commented in this work, but it lowers the cost without compromising the correctness.

Another way of decrease the expected number of polynomial evaluations is to start the procedure with loose estimates of the parameters involved. In case that this produces an error, the calculation must be rerun using the theoretical values assuring us correctness. Related to this, the fact that the bound of the cost involves a global property of the polynomial (like the Lipschitz constant of the polynomial) has been viewed as a handicap of
geometrical algorithms [16]. The same technique that with the parameters of the procedure can be used with the value of the global properties: start with a loose estimation and strength it in case of error.

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users’ Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] Dario Bini and Victor Y. Pan. *Polynomial and matrix computations. Vol. 1*. Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1994. Fundamental algorithms.

[3] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp.

[4] R.M. Burstall. Proving properties of programs by structural induction. *The Computer Journal*, 12(1):41–48, 1969.

[5] W. Forster. Some computational methods for systems of nonlinear equations and systems of polynomial equations. *J. Global Optim.*, 2(4):317–356, 1992.
[6] Steven Fortune. An iterated eigenvalue algorithm for approximating roots of univariate polynomials. *J. Symbolic Comput.*, 33(5):627–646, 2002. Computer algebra (London, ON, 2001).

[7] J.L. García Zapata and J.C Díaz Martín. A geometric algorithm for winding number computation with complexity analysis. *Journal of Complexity*, 28:320–345, 2012.

[8] Juan-Luis García Zapata, Juan Díaz Martín, and Pedro Gómez Vilda. Parallel root-finding method for lpc analysis of speech. In Petr Sojka, Ivan Kopecek, and Karel Pala, editors, *Text, Speech and Dialogue*, volume 3206 of *Lecture Notes in Computer Science*, pages 529–536. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30120-2.

[9] Xavier Gourdon. Algorithmique du theoreme fondamental de l’algebre. Technical report, Institut national de recherche en informatique et en automatique (INRIA), Unité de recherche de Rocquencourt, 1993.

[10] Peter Henrici. Methods of search for solving polynomial equations. *J. Assoc. Comput. Mach.*, 17:273–283, 1970.

[11] Peter Henrici. *Applied and computational complex analysis. Vol. 1*. Wiley Classics Library. John Wiley & Sons Inc., New York, 1988. Power series—integration—conformal mapping—location of zeros, Reprint of the 1974 original, A Wiley-Interscience Publication.

[12] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2nd edition, November 2000.
[13] Bahman Kalantari. *Polynomial root-finding and polynomiography.* World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2009.

[14] Donald E. Knuth. *The art of computer programming. Vol. 2.* Addison-Wesley Publishing Co., Reading, Mass., second edition, 1981. Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing.

[15] A. N. Kolmogorov and S. V. Fomin. *Introductory real analysis.* Dover Publications Inc., New York, 1975. Translated from the second Russian edition and edited by Richard A. Silverman, Corrected reprinting.

[16] Peter Kravanja and Marc Van Barel. *Computing the Zeros of Analytic Functions,* volume 1727 of *Lecture Notes in Mathematics.* Springer, 2000.

[17] D. H. Lehmer. A machine method for solving polynomial equations. *J. ACM,* 8:151–162, April 1961.

[18] C. Andrew Neff and John H. Reif. An efficient algorithm for the complex roots problem. *J. Complexity,* 12(2):81–115, 1996.

[19] Alan V. Oppenheim, Alan S. Willsky, and With S. Hamid. *Signals and Systems (International Edition).* Pearson Education, 2 edition, August 1996.

[20] V. Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Comput. Math. Appl.,* 31(12):97–138, 1996.
[21] Victor Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Rev.*, 39(2):187–220, 1997.

[22] Victor Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *J. Symbolic Computation*, 33:2002, 2001.

[23] Anthony Ralston and Philip Rabinowitz. *A first course in numerical analysis*. McGraw-Hill Book Co., New York, second edition, 1978. International Series in Pure and Applied Mathematics.

[24] James Renegar. On the worst-case arithmetic complexity of approximating zeros of polynomials. *J. Complexity*, 3(2):90–113, 1987.

[25] Arnold Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Mathematisches Institut Universität Tübingen, 1982.

[26] G.A. Sitton, C.S. Burrus, J.W. Fox, and S. Treitel. Factoring very-high-degree polynomials. *Signal Processing Magazine, IEEE*, 20(6):27 – 42, nov. 2003.

[27] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix eigensystem routines—EISPACK guide*. Springer-Verlag, Berlin, second edition, 1976. Lecture Notes in Computer Science, Vol. 6.

[28] J. F. Traub and H. Woźniakowski. Convergence and complexity of Newton iteration for operator equations. *J. Assoc. Comput. Mach.*, 26(2):250–258, 1979.
[29] Paul Van Dooren. Some numerical challenges in control theory. In *Linear algebra for control theory*, volume 62 of *IMA Vol. Math. Appl.*, pages 177–189. Springer, New York, 1994.

[30] J. H. Wilkinson. *The algebraic eigenvalue problem*. Clarendon Press, Oxford, 1965.

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

[32] Xingren Ying and I. Norman Katz. A reliable argument principle algorithm to find the number of zeros of an analytic function in a bounded domain. *Numer. Math.*, 53(1-2):143–163, 1988.