On the Geometry of Graeffe Iteration*

Gregorio Malajovich
Dep. de Matemática Aplicada, Universidade Federal do Rio de Janeiro
Caixa Postal 68530, Rio de Janeiro, RJ, 21945 – BRASIL
gregorio@labma.ufrj.br, http://www.labma.ufrj.br/~gregorio

Jorge P. Zubelli
IMPA
Estrada Dona Castorina 110, Rio de Janeiro, RJ, 22460-320, BRASIL
zubelli@impa.br, http://www.impa.br/~zubelli

Third Revision, April 13, 2000

Abstract

A new version of the Graeffe's algorithms for finding all the roots of univariate complex polynomials is proposed. It is obtained from the classical algorithm by a process analogous to renormalization of dynamical systems.

This iteration is called Renormalized Graeffe Iteration. It is globally convergent, with probability 1. All quantities involved in the computation are bounded, once the initial polynomial is given (with probability 1). This implies remarkable stability properties for the new algorithm, thus overcoming known limitations of the classical Graeffe algorithm.

If we start with a degree-d polynomial, each renormalized Graeffe iteration costs $\mathcal{O}(d^2)$ arithmetic operations, with memory $\mathcal{O}(d)$.

A probabilistic global complexity bound is given. The case of univariate real polynomials is briefly discussed.

A numerical implementation of the algorithm presented herein allowed us to solve random polynomials of degree up to 1000.

1 Introduction

Graeffe Iteration was one of the most prestigious XIX century algorithm for finding roots of polynomials. At that time, computations were performed by hand, by people payed specifically to perform those computations. They were called *calculateurs* [35] or *computers* [21].

*Accepted, J. of Complexity

Let f be a univariate polynomial, of degree d. Its Graeffe iterate is defined as:

$$Gf(x) = (-1)^d f(\sqrt{x}) f(-\sqrt{x}) \ . \tag{1}$$

This defines a many-to-one mapping in the space of all degree-d polynomials (real or complex, as wish). The effect of this mapping is to square each root of f.

After a few Graeffe iterations, the roots of $G^k f$ have (hopefully) incommensurate moduli. This is not true for complex-conjugate roots, which can be worked out in a different way.

Let us momentarily assume, for simplicity, that f is a complex polynomial, with no two roots of the same modulus. Then, $G^k f$ can be written

$$G^k f(x) = \sum_{i=0}^{d} a_i^{(k)} x^i \ , \tag{2}$$

where $a_i^{(k)}$ is given by the $(d-i)$-th symmetric function of the roots of $G^k f$. Therefore, $a_i^{(k)}$ is dominated by:

$$(-1)^{d-i} {\zeta_1}^{2^k} {\zeta_2}^{2^k} \dots {\zeta_{d-i}}^{2^k} \ , \tag{3}$$

where $\zeta_1, \dots, \zeta_d$ are the roots of f ordered with decreasing modulus. A more rigorous version of this claim will appear in Section 5.

Therefore, $-a_{i-1}^{(k)}/a_i^{(k)}$ is a good approximation for ${\zeta_{d-i+1}}^{2^k}$.

Hence it is computationally easy to approximate $|\zeta_i|$ for all i. Although we also obtain $\arg \zeta_i \mod 2^{1-k}\pi$, we will discard this information in this paper to avoid additional complications. There are many classical algorithms to recover the actual value of ζ_i. We deal with this issue in [29]. See also Pan [39] for a discussion.

In this note, we apply a suitable non-uniform change of coordinates (renormalization) to the Graeffe iteration operator, to make it 'convergent' with probability 1.

The algorithm obtained by this change of coordinates will be called *Renormalized Graeffe Iteration.* We use the following systems of coordinates for each iterate $G^k f$ of the Graeffe method applied to f: The coefficients of $G^k(f)$ and all related intermediate computations will be represented in *scaled polar coordinates,* where a complex number w is represented by 'magnitude' $2^{-k} \log_2 |w|$ and 'argument' $\arg w \in [-\pi, \pi]$. Calculations will always be performed 'in coordinates'. The 'magnitude' variables of $G^k f$ 'in coordinates' will converge with probability 1 (Theorem 1 below).

The precise construction of the Renormalized Graeffe Operator is postponed to Section 2.

We also claim that the Renormalized Graeffe algorithm compares well with available numerical software or theoretical algorithms. Also, the authors personally believe that the clean mathematical structure of Graeffe iteration provides a suitable starting point for complexity analysis and algorithm enhancement.

However, in this paper, we are considering a modified problem: our algorithm is designed to find the absolute values of the roots, not the actual roots. Therefore we will compare its complexity to the complexity of finding the absolute value of the roots by other existing algorithms.

In [29], we explain how to modify this algorithm to obtain the actual roots, without endangering the complexity estimates.

The algorithm presented here has arithmetic complexity $\mathcal{O}(d^2)$ for each iteration, and memory size $\mathcal{O}(d)$, where d is the degree of the polynomial.

Complexity will be bounded also in terms of a probability of failure (Theorem 2 below). Our bound improves previous probabilistic bounds (in the sense of probability of success) on the complexity of solving polynomials approximately (See Renegar [42] and Shub-Smale [46]). Usually, probabilistic algorithms assume a certain distribution of the coefficients of the input polynomial, and have a complexity that depends on the degree, the desired accuracy, and a probability of failure δ. (They may also have several other numerical properties that make them suitable for numerical implementation.) For the benefit of the present discussion, let us fix the degree of the input polynomial and the desired accuracy of the solution. Below, we will give a complexity bound of $\mathcal{O}(\log \delta^{-1})$. This can be compared to the bound of $\mathcal{O}(\delta^{-1})$ in [46], for instance. Or to a bound of $O(\log \delta^{-1})$ given in [42] for another probability distribution.

Also, our experiments indicate that our algorithm compares well with practical software, like for instance the algorithm in Matlab (running time $\mathcal{O}(d^3)$ per iteration and memory $\mathcal{O}(d^2)$). Our program could accurately solve polynomials of much larger degree (up to 1000 in some of our experiments, see Section 6).

An informal comparison with existing deterministic algorithms is also possible. The main point of this discussion is to show that our algorithm will not be trivially overrun by a known deterministic algorithm. Therefore, the comparison should be carried on in our probabilistic setting. Also, we are not ruling out the possibility that some of the deterministic algorithms cited below may have a better complexity in a probabilistic setting. Rather, we are comparing known results. Some of the evidence below is conjectural, and relies upon our experiments.

A detailed round-off analysis of renormalized Graeffe iteration will not be discussed in this paper. Instead, some preliminary numerical results are presented in Section 6. They support the empirical fact that typical random degree 1000 polynomials can be solved (in the sense of [7, Section 8.1]) within a precision of 64 bits of mantissa (IEEE 854 double-extended). We expect a factor of a polynomial in $\log d$ bits of mantissa to be sufficient in general, with probability 1. (This is a conjecture).

Formally, probabilistic complexity estimates may be obtained for all "reasonable" probability distributions on the space of polynomials. By reasonable, we mean all probabilities with bounded Radon-Nikodym derivative with respect to Lebesgue probability in the projectivization of coefficient space.

Theorem 1. *There is a renormalization of the Graeffe iteration, such that if f is a degree-d polynomial (in a measure theoretical sense) then with probability 1 this renormalized Graeffe iteration produces $d+1$ sequences, each one converging to some h_i, s.t. $\log|\zeta_i| = h_i - h_{i+1}$, and $\zeta_1, \cdots, \zeta_d$ are roots of f. Moreover, each iteration can be performed in $\mathcal{O}(d^2)$ arithmetical operations and all iterations can be performed with memory $\mathcal{O}(d)$.*

This theorem is constructive, in the sense that an explicit construction of the renormalized Graeffe iteration will be given.

In Section 2, we discuss the precise meaning of renormalization in our context. Its main consequence will be to produce an algorithm operating on a bounded set of numbers. This solves the main stability problem of classical Graeffe iteration that prevented it from finding all roots at once. See for example Henrici's comments on FFT-based Graeffe iteration [18] (Vol III, last paragraph of p. 69).

The definition of renormalization will outlaw FFT-based Graeffe iteration. Indeed, although FFT is known to be stable with respect to vector norms, it is not component-wise stable with respect to the relative error. This means that some of the coefficients of the k-th Graeffe iterate may have a large relative error, and hence some of the roots will be extremely inaccurate. This may be disastrous if one wants to retrieve all the roots at the same time.

Theorem 2. *Let f be a random complex polynomial of degree d. Let $b \geq 1 + \log_2 d$. Then, with probability $1-\delta$, k steps of the Renormalized Graeffe Iteration will approximate the $\log|\zeta_i|$'s with relative precision 2^{-b}, where*

$$k \geq c_1 + c_2 \log_2 b - c_3 \log_2 \delta$$

and c_2, c_3 are universal constants. The constant c_1 depends on the choice of the probability distribution, and on d.

Whenever speaking of random polynomials, we like to consider the normally invariant probability density introduced by Kostlan [25, 26] (See also Section 4). However, the above mentioned result is true for any reasonable probability distribution.

The experimental results in Figure 3 support the conjecture that, under Kostlan's probability distribution, we can fix

$$c_1 = c_4 \log_2 d \ ,$$

where $c_4 \approx 2$ is a universal constant.

We will briefly discuss the real case, and how to deal with complex-conjugate roots or roots with same modulus in Section 5.

Historical remarks. The Graeffe iteration was developed independently by Dandelin (1826), by Graeffe (1837) and by Lobachevsky (1834). We call it

Graeffe iteration to conform with most of the literature. See Householder [20] for early references and priority questions. See Dedieu [9] for an application of Graeffe 's algorithm.

Important theoretical results were obtained by Ostrowskii [35] in 1940. Also, by that time, numerical analysis books mentioned Graeffe iteration as the preferred algorithm for zero-finding (see e.g. Uspensky [52] Page 318. For another early computer implemented algorithm, see Bareiss [1, 2] and also Blish and Curry [6]).

With the advent of digital computing, the practical use of Graeffe iteration seems to have been forgotten.

Most popular zero-finding algorithms seem to be based now on QR iteration (Matlab) or in a several steps, root-finding plus deflation scheme. (e.g. Jenkins and Traub [22]). See, however, Cardinal [8], Edelman and Murakami [13], Emiris, Galligo and Lombardi [14], and Toh and Trefethen [51].

In a more theoretical perspective, Graeffe iteration is used as as a sort of preconditioning for polynomial splitting. Splitting a polynomial means factorizing it into one factor with large roots, and another with small roots. Splitting is used to obtain extremely fast theoretical algorithms (see Schönhage [49, 50], Kirrinnis [23], Neff and Reif [33], Bini and Pan [5], Mourrain and Pan [32], Pan [37, 38, 39], Pan et al. [40], Malajovich and Zubelli [28]). The main practical difficulty for those algorithms seems to be the large precision required by Graeffe iteration.

An important paper by Grau in 1963 [17] laid some of the bases for a version of Graeffe iteration adapted to digital computers. He identified the problem of the increasing *numerical range.* During Graeffe iteration, some of the coefficients can become so large that the floating point system cannot accommodate them anymore.

Most of the literature suggests to find one root at a time and then use deflation. This may suffer from stability problems, as explained for example in [18]. Grau proposed a globally convergent algorithm. Grau's algorithm would involve only bounded quantities.

As far as we know, that paper was completely forgotten. The algorithm suggested by Grau has complexity $\mathcal{O}(d^2)$ and memory usage of $\mathcal{O}(d^2)$. It may be considered as the precursor of the one we shall introduce below.

2 Iterative Algorithms and Renormalization

In this paper, we will produce a version of Graeffe iteration that has *bounded* numerical range, for most input polynomials. The crucial concept in the construction of this algorithm is the idea of *renormalization.*

Renormalization is a tool used in understanding the qualitative behavior of iterative phenomena that range over different scales. A rich theory of renormalization exists for one-dimensional dynamical systems. See Feigenbaum [15], McMullen [30], and De Melo-Strien [31]. As for the multi-dimensional case see Palis-Takens [36].

Before defining what we mean by a renormalized algorithm, we will briefly introduce a notion of *iterative algorithm*. We will adopt as a reference the framework of [7], but we will modify their model to be suitable to our context.

An *iterative algorithm* is not an *algorithm* in the classical sense. While classical algorithms are guaranteed to stop, what is important about iterative algorithms are their asymptotic properties. In order to speak of convergence, we need iterative algorithms to produce an infinite sequence of results, converging to the correct solution. (The reader of [7] may wonder if an *algorithm* is always an *iterative algorithm*. Actually, a *finite-dimensional machine* over $\mathbb{R}$ in the sense of [7] and with integer coefficients may be embedded into an *iterative algorithm* as defined below).

Definition 1. An iterative algorithm M is a Blum-Shub-Smale (BSS) machine over $\mathbb{R}$ with integer coefficients, modified as follows:

1. If the input is in $\mathbb{R}^l \times \mathbb{T}^m$ for a pair (l, m), then the output is in $\mathbb{R}^l \times \mathbb{T}^m$. The integer $l+m$ is called the input size. If the input is denoted by x, the output is denoted $M(x)$, and M^k means the composition $M \circ M \circ \cdots \circ M$

   Also, we say that the iterative algorithm 'computes' the function $x \mapsto \lim_{k\to\infty} M^k(x)$

2. Computation nodes are allowed to perform the following elementary (real) functions: $(+, -, *, /, \exp, \log, \sin, \cos, \mathrm{abs}, \arctan)$.

3. $M_\epsilon(x)$ will denote the result of approximating $M(x)$ by allowing each operation with non-integer parameters to be performed with relative precision ϵ. (This is also known in numerical analysis as the $(1+\epsilon)$ property.) The approximate operations shall be computed by a prescribed algorithm and may be denoted as $+_\epsilon$, $-_\epsilon$, etc... The parameter ϵ is allowed to vary in $(0, \frac{1}{2})$. The machine is supposed to "know" ϵ. This means that the value of ϵ can be used in intermediate calculations. Also, the approximation is performed by some prescribed algorithm. (e.g., IEEE arithmetic with $-\log_2 \epsilon$ bits of mantissa).

4. For all ϵ, the arithmetic complexity of M applied to the input x is the number of elementary function evaluations (from the list of item 1) and branchings performed with input x. We require the arithmetic complexity of the approximate algorithm M_ϵ applied to the input x to be the same as the arithmetic complexity of the original algorithm M applied to input x.

Remark 1. Item 1 will allow us to distinguish between real and angular variables, the latter ones being defined modulo 2π. This will simplify notation when we speak of the distance between points in $\mathbb{R}^l \times \mathbb{T}^m$. Let d be that distance.

Remark 2. It could be argued that the outcome of a branching node would not be well-defined in the presence of numerical error. This is not true in the definition above. The branching nodes of the machine M can be assumed, without loss of generality, to branch on queries of the form $y > 0$ or $y \geq 0$ or $y = 0$. When the machine M gets replaced by M_ϵ, the branching nodes stay formally the same. The value of y, however, is contaminated with a certain numerical error. It is still a perfectly defined real number, and it may be compared to zero. Thus, the branching nodes branch 'correctly', for a slightly perturbed input.

This would lead to disastrous results if an approximate machine enters a 'loop' due to numerical errors. Item 4 in the definition above is there to preclude that sort of loop, by ensuring that the arithmetic complexity of each iteration does not depend on the working precision.

Remark 3. Under this model, the cost of evaluating a degree d polynomial will be $O(d)$.

Remark 4. We should stress that this is a mathematical definition and that we disregard practical matters such as overflow and underflow. We will define later a class of renormalized algorithms, where overflow and underflow will not happen.

Condition 4 is realistic and can be checked in many well-known iterative algorithms. For instance, under certain mild assumptions Newton's iteration satisfies Condition 4. For example, this is the case if the function and its derivatives can be evaluated in a finite number of arithmetic operations independently of the accuracy, and that the linear system is solved in a finite number of arithmetic operations regardless of the accuracy.

Definition 2. We shall say that a function φ can be computed in *finite time* if, and only if, there is a BSS machine over $\mathbb{R}$ modified as above items 2, 3 and 4, that computes φ.

One consequence of the previous definition is the following: suppose a certain function φ can be computed in finite time. Then, its branching set is given by a finite set of equations. Outside the branching set, φ can be written (locally) as a composition of elementary functions. Moreover, only a finite number of such compositions may appear, one corresponding to each set of possible branchings.

A few definitions are in order now. In the sequel we will need to use algorithms depending effectively on a parameter $k \in \mathbb{N}$. Those will be given by a machine M with two inputs, say k and f. However, we will denote the output as $M_k(f)$ and we will write M_k for each of the input-output mappings obtained by restricting that machine to some fixed value of k. Also, in that situation we will speak explicitly of the algorithm $(M_k)_{k \in \mathbb{N}}$ or M_k for short.

The sequence $(M_k)_{k=1}^{\infty}$ can be considered as a sequence of mappings from $\mathbb{R}^l \times \mathbb{T}^m$ into itself. The orbit of f by the sequence $(M_k)_{k=1}^{\infty}$ is the set

$$\mathrm{orb} f \stackrel{\mathrm{def}}{=} \{f, M_1(f), M_2 \circ M_1(f), \dots\} \subset \mathbb{R}^l \times \mathbb{T}^m \ .$$

This set should be understood as the orbit of f by the non-autonomous dynamical system M_k (Not the semi-group !). The closure of orb$f$ will be denoted by $\overline{\mathrm{orb} f}$.

We shall say that a subset of $\mathbb{R}^l \times \mathbb{T}^m$ has full-measure if its complement is contained in a set of null measure. We say that some property is true almost everywhere (a.e.) if that property is true in a full-measure set. We can now define a *renormalized algorithm* and make precise our concept of renormalization:

Definition 3. The algorithm $(M_k)_{k\in\mathbb{N}}$ (or M_k for short) is said to be a *renormalized iterative algorithm to compute* $\varphi : \mathbb{R}^l \times \mathbb{T}^m \to \mathbb{R}^s$, $f \mapsto \varphi(f)$, where $0 \leq s \leq l$, if, and only if, it satisfies the Axioms 1 through 4 below.

Axiom 1 (Consistency). For almost every $f \in \mathbb{R}^l \times \mathbb{T}^m$ we have

$$\lim_{k\to\infty} \left(\pi \circ M_k \circ M_{k-1} \circ \cdots \circ M_1\right)(f) = \varphi(f) \ ,$$

where π is the projection of $\mathbb{R}^l \times \mathbb{T}^m$ onto the first s coordinates of $\mathbb{R}^l$.

Axiom 2 (Arithmetic Complexity). The arithmetic complexity of M_k with input f is bounded in terms of the size $l+m$ of f, independently of k and the coefficients of f.

Axiom 3 (Propagation). For almost every $f \in \mathbb{R}^l \times \mathbb{T}^m$, there exists a compact neighborhood $V \subset \mathbb{R}^l \times \mathbb{T}^m$ of orb$f$ (under $\{M_k\}$) and C such that $M_k|_V$ is Lipschitz with constant C_k and eventually $C_k < C$.

Axiom 4 (Stability). For almost every $f \in \mathbb{R}^l \times \mathbb{T}^m$, there exists a compact neighborhood $V \subset \mathbb{R}^l \times \mathbb{T}^m$ of orb$f$ (under $\{M_k\}$) and $B \in \mathbb{R}$ such that $\forall g \in V$ and $\forall k$,

$$\mathrm{d}\left(M_{k,\epsilon}(g) \ , \ M_k(g)\right) < \epsilon B \ .$$

Our concept of renormalized iterative algorithm subsumes several 'reasonable' properties of iterative algorithms. Axiom 1 allows our algorithm to carry more information than what is actually required at output. Yet, we want our algorithm to produce a sequence converging to the expected result, for almost every input. Axioms 3 and 4 will rule out unstable algorithms. The idea behind Axiom 2 is that any honest iterative algorithm should have bounded arithmetic complexity for each iteration. This prevents the use of multiple precision arithmetic to obtain stability at the expense of (possibly exponentially many) extra arithmetic operations.

We shall now explore some consequences of the definition of renormalized iterative algorithm we proposed above. More specifically, our first goal is to obtain a (non-uniform) error bound on the result of iterating k times a renormalized algorithm with precision ϵ.

Lemma 1. *Let M be a renormalized iterative algorithm . Then, for almost every f and for each k, we have that for sufficiently small $\epsilon > 0$*

$$\mathrm{d}\left(M_{k,\epsilon} \circ \cdots \circ M_{1,\epsilon}(f), M_k \circ \cdots \circ M_1(f)\right) < kA^k B\epsilon \ .$$

Here, A and B, depend on f, but not on k. In particular, that will be true for values of ϵ of the form

$$\epsilon \leq \frac{\rho}{kA^k B} \ ,$$

where ρ is defined below.

As an immediate consequence of Lemma 1 we have that

$$\| (\pi \circ M_{k,\epsilon} \circ \cdots \circ M_{1,\epsilon})(f) - (\pi \circ M_k \circ \cdots \circ M_1)(f)) \|_2 < kA^k B\epsilon \ .$$

Proof of Lemma 1. We now describe the construction of the constants in the statement of Lemma 1. Combining Axioms 3 and 4, there exists a compact neighborhood W of $\mathrm{orb} f$ such that:

1. Every mapping M_k is Lipschitz on W. Moreover, there exists a constant A such that for every k, the norm of the Lipschitz constant of M_k is uniformly bounded by $A = \max(1, \sup_k C_k)$, C_k as in Axiom 3.

2. There exists ϵ_0 such that $\forall \epsilon < \epsilon_0$ we have

   $$\mathrm{d}(M_{k,\epsilon}(g), M_k(g)) < \epsilon B \ , \qquad \forall g \in W \ .$$

   We then define ρ as the minimum of ϵ_0 and the distance of $\mathrm{orb} f$ to the boundary of W.

We may now conclude the proof of Lemma 1. Set

$$g = (M_{k-1} \circ \cdots \circ M_1)(f) \ ,$$

and

$$h = (M_{k-1,\epsilon} \circ \cdots \circ M_{1,\epsilon})(f) \ .$$

In that case, we want to bound

$$\mathrm{d}(M_{k,\epsilon}(h), M_k(g)) \leq \mathrm{d}(M_{k,\epsilon}(h), M_k(h)) + \mathrm{d}(M_k(h), M_k(g)) \ .$$

If $h \in W$, one can bound

$$\mathrm{d}(M_k(h), M_{k,\epsilon}(h)) \leq B\epsilon$$

and

$$\mathrm{d}(M_k(h), M_k(g)) \leq A \ \mathrm{d}(g, h) \ .$$

By induction,

$$\mathrm{d}(g,h) \le (k-1)A^{k-1}B\epsilon \ ,$$

and hence

$$\mathrm{d}(M_{k,\epsilon}(h), M_k(g)) \le kA^kB\epsilon \ .$$

The condition

$$\epsilon < \frac{\rho}{kA^kB}$$

guarantees that $h \in W$. □

If we use a Turing machine model (or any other classical discrete complexity model), we can perform all the operations of M in finite precision, and obtain:

Lemma 2. *Let M be a renormalized iterative algorithm. For almost every f, assume that the truncation error is bounded by*

$$\| \left(\pi \circ M_k \circ \cdots \circ M_1\right)(f) - \varphi(f) \|_2 < E^{2^k} \ ,$$

where $E = E(f) \in (0,1)$. Then, the complexity of approximating $\varphi(f)$ with precision δ is $\mathcal{O}((\log_2 \frac{1}{\delta})^{1+\alpha})$ where $\alpha > 0$ is arbitrarily small.

It is assumed above that the cost of arithmetic with l bits of mantissa is $O(l^{1+\alpha})$, for all $\alpha > 0$. See [4] pp 78-79 for a sharper bound on the complexity of long integer multiplication.

Proof of Lemma 2. Let

$$k = \left\lceil \log_2 \frac{-\log_2 \frac{\delta}{2}}{-\log_2 E} \right\rceil \ ,$$

so that

$$E^{2^k} \le \frac{\delta}{2} \ .$$

Choose ϵ so that Lemma 1 holds, i.e., $\epsilon < \rho / kA^kB$, and such that $kA^kB\epsilon < \delta/2$. This can be done for

$$\epsilon < c_1 \delta (\log \delta^{-1})^{c_2}$$

for some constants c_1 and c_2 dependent of f. The total cost of computing M_k is $\mathcal{O}\left(a(\log_2 \epsilon^{-1})^{1+\alpha}\right)$, where a is the arithmetic complexity of the algorithm and α is arbitrarily small. Since f is fixed, a is a constant and $k < \log_2 \log_2 \delta^{-1}$ the total cost is

$$\mathcal{O}\left((\log_2 \delta^{-1})^{1+\alpha}\right)$$

for all α arbitrarily small. □

The complexity bound of Lemma 2 is non-uniform in f. It is also non-effective, in the sense that we give no procedure to estimate k and ϵ without the knowledge of ρ, B and A. Indeed, those quantities may depend on f. It would be of some interest to bound those quantities in a probabilistic setting, similar to Theorem 2.

Definition 4. Let M be an iterative algorithm to compute $\Phi : F \mapsto \Phi(F)$, i.e., $\lim_{n\to\infty} M^n(F) = \Phi(F)$). We say that M_k is a renormalization of M if

1. M_k is a renormalized iterative algorithm to compute $\varphi(f)$.
2. There are functions ψ and η, defined almost everywhere and computable in finite time, such that the diagram

$$\begin{array}{ccc} F & \xrightarrow{\psi} & \psi(F) \\ {\scriptstyle\Phi}\big\downarrow & & \big\downarrow{\scriptstyle\varphi} \\ \Phi(F) & \xleftarrow{\eta} & \varphi(\psi(F)) \end{array}$$

   commutes.
3. There is a function R, computable in finite time, so that the diagram

$$\begin{array}{ccc} F & \xrightarrow{R^k\circ\psi} & R^k(\psi(F)) \\ {\scriptstyle M}\big\downarrow & & \big\downarrow{\scriptstyle M_k} \\ M(F) & \xrightarrow{R^{k+1}\circ\psi} & R^{k+1}(\psi(M(F))) \end{array}$$

   commutes for all k.

Theorem 1 can be stated now in a more concise way: Let $\zeta : f \mapsto \zeta(f)$ be the function that associates, to any univariate degree d polynomial f, its roots $\zeta_1, \ldots, \zeta_d$ ordered by decreasing modulus. We have the following:

Theorem. *There is a renormalization of the Graeffe iteration to compute* $|\zeta(f)| = (|\zeta_1(f)|, \ldots, |\zeta_d(f)|)$. *Each iteration has arithmetic complexity* $\mathcal{O}(d^2)$ *and uses memory* $\mathcal{O}(d)$.

3 Recurrence Relations and the Renormalization of Graeffe

It is time to construct the renormalized Graeffe iteration. Let $f(x) = f_0 + f_1 x + \cdots + f_d x^d$. Let $h = Gf$ be its Graeffe iterate. The coefficients of h can be written as

$$h_i = (-1)^d \sum_{\substack{0\le i-j\le d\\ 0\le i+j\le d}} (-1)^{i-j} f_{i-j} f_{i+j} \ . \qquad (4)$$

For convenience, we rewrite (4) as

$$h_i = (-1)^{d+i} {f_i}^2 + 2 \sum_{1 \le j \le \min(i,d-i)} (-1)^{d+i-j} f_{i-j} f_{i+j} \ . \tag{5}$$

The next step is to write those equations in terms of the log of the coefficients. More precisely, we will have to deal with the following two quantities:

$$f_i^{\log} \stackrel{\text{def}}{=} \log |f_i| \tag{6}$$

and

$$f_i^{\arg} \stackrel{\text{def}}{=} \arg f_i \ . \tag{7}$$

It is possible now to construct the renormalized Graeffe iteration G_k. This iteration will map $2^{-k} f^{\log}$ and $f^{\arg}$ into $2^{-k-1} h^{\log}$ and $h^{\arg}$.

For that purpose, we introduce the notation:

$$\left\{ \begin{array}{lcl} f^k & \stackrel{\text{def}}{=} & (2^{-k} f^{\log}, f^{\arg}) \\ h^{k+1} & \stackrel{\text{def}}{=} & (2^{-k-1} h^{\log}, h^{\arg}) \end{array} \right. \ . \tag{8}$$

We also introduce operators:

$$(x, \alpha) \boxtimes (y, \beta) \stackrel{\text{def}}{=} (x + y, \alpha + \beta) \tag{9}$$

$$(x, \alpha) \boxed{\scriptstyle \lambda} \stackrel{\text{def}}{=} (\lambda x, \lambda \alpha) \tag{10}$$

$$\boxed{\scriptstyle z}_k (x, \alpha) \stackrel{\text{def}}{=} (x + 2^{-k} \log |z|, \alpha + \arg z) \tag{11}$$

and

$$(x, \alpha) \boxplus_k (y, \beta) \stackrel{\text{def}}{=} \left(2^{-k} \log \left| e^{i\alpha + 2^k x} + e^{i\beta + 2^k y} \right|, \arg \left(e^{i\alpha + 2^k x} + e^{i\beta + 2^k y} \right) \right) \tag{12}$$

We remark that the purpose of the sub-index k in the above formulae is to keep track of the degree of the renormalization. For operations which do not change, we omitted the sub-index. The operator $\boxed{\scriptstyle z}_k$ stands for the multiplication of a renormalized value by a (non-renormalized) constant z.

Also, binary renormalized operations are defined for operands with the same renormalization index. Therefore, one should first convert f_i^k to f_i^{k+1} before attempting to 'multiply' it with a factor of renormalization index of order $k+1$. This conversion will be implicit in the formulae below.

Equation (5) becomes:

$${h_i}^{k+1} = \left(\boxed{\scriptstyle s_{i0}}_{k+1} \left({f_i}^k \right) \boxed{\scriptstyle 2} \right) \boxplus_{k+1} \left(\boxed{\scriptstyle 2}_{k+1} \ \boxed{+}_{j=1}^{\min(i,d-i)}{}_{k+1} \left(\boxed{\scriptstyle s_{ij}}_{k+1} {f_{i-j}}^k \boxtimes {f_{i+j}}^k \right) \right) \ , \tag{13}$$

where $s_{ij} = (-1)^{d+i-j}$. Recall that above, k is a superscript, not an exponent. The 'renormalized operations' above are easy to implement in terms of the classical ones. The most delicate being the renormalized sum. We give here our preferred algorithm:

Example 1. How to compute the 'Renormalized sum':

$$(c, \gamma) := (a, \alpha) \boxplus_k (b, \beta)$$

```
If a > b, do:
        s = exp iα + exp (iβ + 2^k (b − a))
        c = a + 2^{−k} ln |s|
        γ = arg(s)
else
        s = exp iβ + exp (iα + 2^k (a − b))
        c = b + 2^{−k} ln |s|
        γ = arg(s)
endif
```

We remark that in the above formula, the complex arithmetic operations can be performed in terms of real elementary ones. Moreover, in numerical implementations if k is large enough, as compared to ϵ, it may be faster to approximate (c, γ) with (a, α) or (b, β), whichever is larger.

In order to finish the proof of Theorem 1, we still need a few remarks about the renormalized Graeffe iteration, which we just constructed. Clearly, in order to compute formula (13), we only need memory space of the order $\mathcal{O}(d)$ and time of the order $\mathcal{O}(d^2)$.

When $k \to \infty$, the quantities $|f_i^k|$ are all convergent.

If G_k is the renormalized Graeffe iteration, we define G_∞ as the limit of G_k when k goes to infinity. This means that renormalized sums $\boxplus_k$ are replaced by their limit $\boxplus_\infty$, where

$$(a, \alpha) \boxplus_\infty (b, \beta) = \begin{cases} (a, \alpha) & \text{if } a > b \\ (b, \beta) & \text{if } a < b \\ \text{undefined} & \text{if } a = b \end{cases} .$$

It is easy to see that $G_k \to G_\infty$ almost everywhere, pointwise and in the C^1 topology. By this we mean that for f almost everywhere,

$$\lim_{k \to \infty} |G_k(f) - G_\infty(f)| + \|DG_k(f) - DG_\infty(f)\|_F = 0 ,$$

where $\|\cdot\|_F$ denotes the Fröbenius norm. In order to avoid a rather tedious calculation, we can establish this fact from the pointwise C^1 convergence almost everywhere of $(a, \alpha) \boxplus_k (b, \beta)$ to $(a, \alpha) \boxplus_\infty (b, \beta)$ and similarly for the other renormalized operations.

We define:

$$\begin{aligned}
\psi(f_0, \cdots, f_d) &= (\log|f_0|, \cdots, \log|f_d|; \arg|f_0|, \cdots, \arg|f_d|) \\
\eta(r_0, \cdots, r_d; a_0, \cdots, a_d) &= (\exp(r_0 - r_1), \cdots, \exp(r_{d-1} - r_d)) \\
R(r_0, \cdots, r_d; a_0, \cdots, a_d) &= \left(\frac{r_0}{2}, \cdots, \frac{r_d}{2}; a_0, \cdots, a_d\right) .
\end{aligned}$$

We define Φ as the function that associates to a polynomial f the values $|\zeta_1|, \ldots, |\zeta_d|$ where ζ_i are the roots of f, ordered by decreasing modulus.

Then, we set

$$\varphi = \eta^{-1} \circ \Phi \circ \psi^{-1} .$$

With the definitions above, G_k is indeed a renormalization of the classical Graeffe algorithm to compute Φ:

Proposition 1. *Renormalized Graeffe Iteration is a renormalized iterative algorithm (in the sense of Definition 3) to compute* Φ*.*

The proof of this proposition will conclude the proof of Theorem 1. In order to establish Proposition 1, we should check that our algorithm, as described in equation (13), satisfies Axioms 1 to 4.

Axiom 1 is verified by construction. Axiom 2 follows from the recurrence formula (13).

The proof that our algorithm satisfies Axiom 3 will require a technical lemma. Before stating it, we recall some notation:

$\mathrm{orb}(f) = \{f; G_1(f); G_2 \circ G_1(f); \ldots; G_k \circ \cdots \circ G_1(f); \ldots\}$ is the orbit of f under the sequence (G_k). Its closure is denoted by $\overline{\mathrm{orb}(f)}$.

We will show the following lemma, which implies satisfaction of Axiom 3.

Lemma 3. *For almost every f and any $\delta > 0$, there exist a compact neighborhood $W \subseteq \mathbb{R}^l \times \mathbb{T}^m$ of $\overline{orb(f)}$ and an integer k_0 such that, G_k is a local diffeomorphism $W \to G_k(W)$, and such that for $k_0 \leq k \leq \infty$, the derivative of $G_k|_W$ is bounded by $2 + \delta$*

Proof of Lemma 3. The proof is divided in several steps.

Step 1: For any $j \in \mathbb{N}$, there is a full-measure set U_j such that $G_j \circ \cdots \circ G_1$ is well-defined, and a local diffeomorphism in U_j. Hence, there is a full-measure set $U_{j,k}$ such that $G_k \circ (G_j \circ \cdots \circ G_1)$ is defined on $U_{j,k}$ and for all $f \in U_{j,k}$ there is a neighborhood V_f of $G_j \circ \cdots \circ G_1(f)$ such that G_k is a diffeomorphism $V_f \to G_k(V_f)$.

Step 2: Let U_∞ be the set of all f such that G_k is a diffeomorphism near $(\pi^{-1} \circ \varphi)(f)$ for all values of k that are large enough. Then U_∞ contains the set of complex polynomials without roots of the same modulus. Hence U_∞ has full measure.

Step 3: Let $U = U_\infty \cap \left(\bigcap_j U_j\right) \cap \left(\bigcap_{jk} U_{jk}\right)$. Then U has full measure. Moreover, let $f \in U$. Then G_k is a local diffeomorphism with derivative of norm $< 2 + \delta/2$ in an open neighborhood $V_k(g)$ of every $g \in \overline{\mathrm{orb} f}$, for all $k \geq k_0(g)$. Indeed, if we write $G_k = G_\infty + (G_k - G_\infty)$, we can make the C^1 norm of the second term arbitrarily small, namely less than $\delta/2$. We know that the norm of the derivative of G_∞ is precisely 2, hence the bound $2 + \delta/2$. In the particular case $\delta = \infty$ we can set $k_0 = 1$.

Step 4: Since $G_k \to G_\infty$ pointwise in the C^1 topology and for g almost everywhere, we can assume that $\bigcap_{k \geq k_0} V_k(g)$ contains an open ball $V(g)$ of center g, where G_k is a local diffeomorphism with derivative bounded by $2 + \delta/2$.

Step 5: Since $\overline{\mathrm{orb}(f)}$ is compact, the union $\bigcup_{g \in \overline{\mathrm{orb}(f)}} V(g)$ has a finite sub-cover $\bigcup_{g \in \Gamma} V(g)$, and we set $W = \bigcup_{g \in \Gamma} \overline{V(g)}$. Then we set $k_0 = \max_{g \in \Gamma} k_0(g)$, and we obtain that for any $k \geq k_0$, G_k is a local diffeomorphism in W, with derivative of norm bounded by $2 + \delta$.

□

The proof that our algorithm satisfies Axiom 4 is divided in two parts, dealing (respectively) with small and large values of k.

Lemma 4. *For almost every f and for all k, there exist an open neighborhood U containing f and $B > 0$ such that for all $g \in U$ and for all ϵ small enough*

$$\mathrm{d}\left(G_{k,\epsilon}(g) - G_k(g)\right) < B\epsilon \ .$$

Lemma 5. *For almost every f, there exist k_0, $B > 0$, and an open neighborhood U containing f, such that for all $g \in U$ and $k \geq k_0$, for all ϵ small enough,*

$$\mathrm{d}\left(G_{k,\epsilon}(g), G_k(g)\right) < B\epsilon \ .$$

Lemmas 4 and 5 together imply that for almost all f, there is a neighborhood U containing f such that for all $g \in U$,

$$\mathrm{d}\left(G_{k,\epsilon}(g), G_k(g)\right) < B\epsilon$$

independently of k.

Since $\mathrm{orb}(f)$ is defined for almost every f and admits a compact neighborhood V, we can select a finite subcover of those neighborhoods, and hence find a finite B valid for all $g \in V$. Therefore, our algorithm satisfies Axiom 4.

Proof of Lemma 4: Our iteration G_k can be written in terms of the following real operations: $+$, $-$, cos, sin, arctan, multiplication by 2^k, by 2^{-k}, absolute value, exp, log.

The set of inputs such that a 'log of zero' or an 'absolute value of zero' occurs has zero measure. Therefore, for almost every f, $G_k(f)$ is computed by a composition of analytic functions. Also, for almost every f, none of the output values is zero.

Therefore, for every intermediate quantity x_l, the derivative of any output y_m with respect to x_l is finite (say $\leq D_{lm}$). Therefore, a relative perturbation of ϵ in x_l leads to a perturbation of size $|x_l| D_{lm} \epsilon$ in y_m. Thus, we set $B = \frac{1}{2} \sum_{l,m} D_{lm} |x_l|$. By continuity, a perturbation of ϵ in each intermediate value x_l leads to a perturbation smaller than $B\epsilon$, for input g in a certain neighborhood of f. $\square$

Proof of Lemma 5: Let W be the set of all f such that $G_\infty(f)$ is well-defined. Recall that

$$\boxplus_\infty : \quad ((a,\alpha),(b,\beta)) \quad \mapsto \quad \begin{array}{ll} (a,\alpha) & \text{if } a > b \\ (b,\beta) & \text{if } b > a \end{array}$$

and that the operator $\boxplus_\infty$ is not defined for $a = b$. Therefore, W is open and has full measure. Let $f \in W$ and U be a small connected neighborhood containing f. Let $g \in U$, then by taking U small enough and k large enough, we can guarantee that $G_k(g)$ and $G_{k,\epsilon}(g)$ are well-defined. Since U is connected, all the branching outcomes in the computation of $G_k(g)$ and $G_{k,\epsilon}$ are the same, hence G_k restricted to U is a composition of locally analytic functions. We can assume without loss of generality that all derivatives are bounded, hence there is a constant B such that

$$\mathrm{d}\left(G_{k,\epsilon}(g), G_k(g)\right) < B\epsilon$$

for ϵ small enough, but still independent of the choice of g.

$\square$

This concludes the proof of Proposition 1 and hence of Theorem 1.

4 Probability of Success

Through this section, $\|.\|_d$ will denote Weyl's unitary invariant norm ([54] III–7, pp 137–140) in the space $\mathcal{P}_d$ of complex polynomials of degree at most d (See [7]). If $f(x) = \sum_{i=0}^{d} f_i x^i$, then

$$\|f\|_d = \sqrt{\sum_{i=0}^{d} \frac{|f_i|^2}{\left(\begin{array}{c} d \\ i \end{array}\right)}} \, .$$

This norm is invariant under the following action of the group $U(2)$ of unitary 2×2 matrices: if $\varphi = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$ is unitary, define

$$f^{\varphi}(y) = (\gamma y + \delta)^d f\left(\frac{\alpha y + \beta}{\gamma y + \delta}\right) \ .$$

Thus, $\|f^{\varphi}\|_d = \|f\|$. See also Ch. 12, Th. 1 of [7].

This is in some sense the most 'natural' norm in the space of all polynomials. More information about that norm, its associated probability distribution and its applications can be found in [3, 10, 11, 25, 27, 28, 43, 44, 45, 46, 47].

Let $\mathbb{P}\mathcal{P}_d$ be the projectivization of normed complex vector space $(\mathcal{P}_d, \|.\|_d)$. We can define the 'sine' distance in $\mathbb{P}\mathcal{P}_d$ by:

$$d_{\mathbb{P}}(f,g) = \min_{\lambda \in \mathbb{C}} \frac{\|f - \lambda g\|}{\|f\|} = \sin \varrho(f,g) \ .$$

where ϱ is the usual Riemann metric in $\mathbb{P}\mathcal{P}_d$. Also, there is a natural volume form in $\mathbb{P}\mathcal{P}_d$. Normal invariant distributed polynomials in $\mathcal{P}_d$ correspond to uniformly distributed 'polynomials' in $\mathbb{P}\mathcal{P}_d$.

Let f be a random polynomial (in any of the two equivalent senses above). Then we may consider its roots $\zeta_1, \cdots, \zeta_d$ as random variables. The joint distribution of $\zeta_1, \cdots, \zeta_d$ was studied by Kostlan in estimates:

Lemma 6. *Let f be random in the sense above. The probability that*

$$\min_{|\zeta_i| > |\zeta_j|} \frac{|\zeta_i|}{|\zeta_j|} > 1 + \epsilon$$

is larger than $1 - M\epsilon$, where M is a positive constant depending on d.

This result is also true if we chose f random with respect to any other probability distribution with bounded Radon-Nikodym derivative with respect to the volume form in $\mathbb{P}\mathcal{P}_d$. The constant M will have to be multiplied by the maximum of the Radon-Nikodym derivative of the new distribution with respect to the volume form.

Lemma 6 will be a consequence of the 'Condition Number Theorem' below. Let

$$\rho(f) = \min_{|\zeta_i| < |\zeta_j|} 1 - \frac{|\zeta_i|}{|\zeta_j|} \ .$$

We will interpret $\rho(f)^{-1}$ as a condition number. Let Σ_G be the locus of ill-posed problems, i.e., the set of polynomials such that $|\zeta_i| = |\zeta_j|$ for some $i \neq j$. Then, we shall prove the following:

Theorem 3 (Condition Number Theorem for Graeffe Iteration).

$$\rho(f) \geq \frac{d_{\mathbb{P}}(f, \Sigma_G)}{\sqrt{d}}$$

Therefore, the probability that $\rho(f) > \epsilon$ is no less than $1 - \mathbf{V}ol\ V_{\epsilon\sqrt{d}}\Sigma_G$ where $V_{\epsilon\sqrt{d}}$ denotes an $\epsilon\sqrt{d}$-neighborhood. This volume is of $O(\epsilon\sqrt{d}) \leq M\epsilon$.

Lemma 7. *Let $f(x) = (x - \zeta_1)g(x) \in \mathcal{P}_d$, where $g \in \mathcal{P}_{d-1}$. Then*

$$\|g\|_{d-1} \leq \sqrt{\frac{d}{1 + |\zeta_1|^2}} \|f\|_d$$

Proof of Lemma 7. We start with the easy case and assume that $\zeta_1 = 0$. Set $g(x) = \sum_{i=0}^{d-1} g_i x^i$. Then, $f(x) = \sum_{i=1}^{d} g_{i-1} x^i$. Now,

$$\|g\|_{d-1}^2 = \sum_{i=0}^{d-1} \frac{|g_i|^2}{\begin{pmatrix} d-1 \\ i \end{pmatrix}} = \sum_{i=1}^{d} \frac{|g_{i-1}|^2}{\begin{pmatrix} d-1 \\ i-1 \end{pmatrix}} = \sum_{i=1}^{d} \frac{d}{i} \frac{|g_{i-1}|^2}{\begin{pmatrix} d \\ i \end{pmatrix}} \leq d\|f\|_d^2$$

and hence $\|g\|_{d-1} \leq \sqrt{d}\|f\|_d$.

For the general case, we will use $U(2)$-invariance of $\|.\|_d$ and $\|.\|_{d-1}$. Let φ be a convenient unitary matrix:

$$\varphi = \frac{1}{\sqrt{1 + |\zeta_1|^2}} \begin{bmatrix} 1 & \zeta_1 \\ -\bar{\zeta_1} & 1 \end{bmatrix} .$$

The choice of φ has the particularity that $f^\varphi(0) = f(\zeta_1) = 0$. We can compute f^φ in terms of g^φ:

$$f^\varphi(y) = \sqrt{1 + |\zeta_1|^2}\ y\ g^\varphi(y) .$$

Using the easy case,

$$\|\sqrt{1 + |\zeta_1|^2} g^\varphi\|_{d-1} \leq \sqrt{d}\|f^\varphi\|_d .$$

By $U(2)$-invariance,

$$\begin{aligned} \|g\|_{d-1} &= \|g^\varphi\|_{d-1} \\ &= \frac{1}{\sqrt{1 + |\zeta_1|^2}} \|\sqrt{1 + |\zeta_1|^2} g^\varphi\|_{d-1} \\ &\leq \frac{\sqrt{d}}{\sqrt{1 + |\zeta_1|^2}} \|f^\varphi\|_d \\ &= \frac{\sqrt{d}}{\sqrt{1 + |\zeta_1|^2}} \|f\|_d . \end{aligned}$$

□

Lemma 8. *Let $g \in \mathcal{P}_{d-1}$. Then $\|g\|_d \leq \|g\|_{d-1}$.*

Proof of Lemma 8.

$$\|g\|_d^2 = \sum_{i=0}^{d-1} \frac{|g_i|^2}{\begin{pmatrix} d \\ i \end{pmatrix}} = \sum_{i=0}^{d-1} \frac{d-i}{d} \frac{|g_i|^2}{\begin{pmatrix} d-1 \\ i \end{pmatrix}} \leq \|g\|_{d-1}^2 \ .$$

□

Putting Lemma 7 and Lemma 8 together,

Lemma 9. *Let $f(x) = (x-\zeta_1)g(x) \in \mathcal{P}_d$, where $g \in \mathcal{P}_{d-1}$. Then*

$$\|g\|_d \leq \frac{\sqrt{d}}{\sqrt{1+|\zeta_1|^2}} \|f\|_d \ .$$

Proof of Theorem 3. Let $f(x) = (x-\zeta_1)(x-\zeta_2)\cdots(x-\zeta_d)$ and order the ζ_i's such that

$$\rho(f) = \min_{|\zeta_i|<|\zeta_j|} 1 - \frac{|\zeta_i|}{|\zeta_j|} = 1 - \frac{|\zeta_2|}{|\zeta_1|} \ .$$

Define $h(z) = (x-\zeta_1(1-\rho))(x-\zeta_2)\cdots(x-\zeta_d) \in \Sigma_G$. Then

$$\begin{aligned} \|f-h\|_d &= \|\zeta_1 \rho (x-\zeta_2)\cdots(x-\zeta_d)\|_d \\ &= |\zeta_1|\rho \|(x-\zeta_2)\cdots(x-\zeta_d)\|_d \\ &\leq |\zeta_1|\rho \frac{\sqrt{d}}{\sqrt{1+|\zeta_1|^2}} \|f\|_d \ . \end{aligned}$$

Hence,

$$\frac{\|f-h\|_d}{\|f\|_d} \leq \frac{|\zeta_1|}{\sqrt{1+|\zeta_1|^2}} \rho\sqrt{d} \leq \rho\sqrt{d} \ .$$

Thus,

$$d_{\mathbb{P}}(f, \Sigma_G) \leq \rho\sqrt{d} \ .$$

□

Proof of Theorem 2. We set $\delta = M\epsilon$, where M is a constant such that the volume of an $\epsilon\sqrt{d}$ neighborhood of Σ_G is less than $M\epsilon$. With probability larger than $1-\delta$,

$$\max_{|\zeta_i|<|\zeta_j|} \frac{|\zeta_i|}{|\zeta_j|} > 1 - \epsilon \ .$$

We now use that if $N > (\log 2)/\epsilon$, then $(1-\epsilon)^N < 1/2$. We set $k_1 = 1 + \lceil \log_2 \epsilon^{-1} \rceil$ and take $N = 2^{k_1}$ in the previous formula, we obtain:

$$\max_{|\zeta_i|<|\zeta_j|} \frac{|\zeta_i|^{2^{k_1}}}{|\zeta_j|^{2^{k_1}}} < (1-\epsilon)^{2^{k_1}} < \frac{1}{2} \ .$$

Furthermore, $1 + \log_2 b$ iterations ensure that

$$\max_{|\zeta_i|<|\zeta_j|} \frac{|\zeta_i|^{2^k}}{|\zeta_j|^{2^k}} < 2^{-1-b} ,$$

provided that

$$k = k_1 + 1 + \log_2 b .$$

Assume we have reordered the roots so that $|\zeta_1| > \cdots > |\zeta_d|$. We claim that we have in this case that the sum

$$\sum_{i_1<\cdots<i_r} \zeta_{i_1}^{2^k} \ldots \zeta_{i_r}^{2^k} = \zeta_1^{2^k} \ldots \zeta_r^{2^k} (1+\varepsilon) ,$$

where

$$|\varepsilon| < 2^{-b} .$$

Indeed, from the ordering it follows that

$$|\zeta_{i_1}^{2^k} \ldots \zeta_{i_r}^{2^k}| < |\zeta_1^{2^k} \ldots \zeta_r^{2^k}| 2^{-(1+b)((i_1-1)+(i_2-2)+\cdots+(i_r-r))} .$$

Now, for r fixed, let

$$\mathcal{S}_l = \{(i_1, \ldots, i_r) \ \mid i_1 < \cdots < i_r \text{ and } i_1 + \ldots i_r - r(r+1)/2 = l\} .$$

Note that $\mathcal{S}_0 = \{(1, \ldots, r)\}$, and $\mathcal{S}_1 = \{(1, \ldots, r-1, r+1)\}$. Also, $\mathcal{S}_2 = \{(1, \ldots, r-1, r+2); (1, \ldots, r-2, r, r+1)\}$

In general, every multi-index $i_1 < \cdots < i_r$ may be obtained by starting from $1 < 2 < \cdots < r$ and increasing one of the indices, in such a way not two indices are equal. Then $\mathcal{S}_k$ is the set of multi-indices obtained after k steps. Therefore, we may bound $\#\mathcal{S}_{k+1} \leq d\ \#\mathcal{S}_k$ to get

$$\sum_{k\geq 1} 2^{-(b+1)k}\ \#\mathcal{S}_k \leq 2^{-b-1}(1 + 2^{-b} + 2^{-b}d + 2^{-2b-1}d^2 \ldots) \leq 2^{-b} ,$$

under the assumption that $d \leq 2^b$. This concludes the proof of Theorem 2. □

5 Newton Diagram Revisited

Following Ostrowskii [35], we introduce Newton diagram, but now with a log scale. The Newton diagram of a polynomial f is the graph of the piecewise linear function defined by $g(i) = -\log|f_i|$. (Our construction differs from Ostrowskii's point of view. He used to define the Newton diagram as the convex hull of that set of points). It makes sense to renormalize Newton diagram, as we did with polynomials $G^k f$.

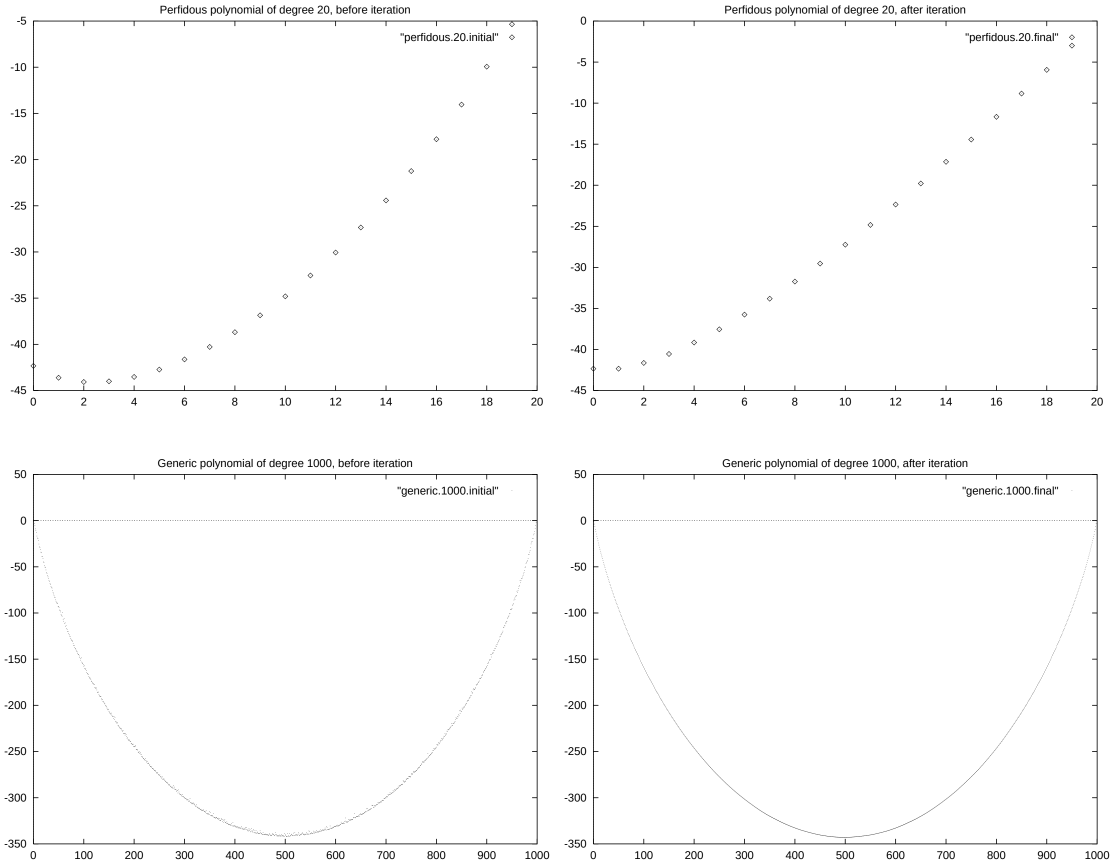

Figure 1: Examples of Newton diagrams

Therefore, we may speak of a Newton diagram *on the limit*. Assume (as we did through this point) that all roots of f have different moduli. Then the derivative of this diagram on the piecewise linear part over $[i, i+1]$ will correspond to $\log|\zeta_{d-i}|$, where ζ_i is the i-th largest root of f in modulus. It follows that the Newton diagram will be the graph of a convex function.

We should consider now the general case. If f has several roots of the same moduli, some of the ratios f_i/f_{i+1} will converge to the ratios of the coefficients of the factor of f containing those roots. The case of three roots of same moduli is illustrative:

Let $\zeta_1, \ldots, \zeta_d$ be arranged by non-increasing moduli, and assume $|\zeta_i| = |\zeta_{i+1}| = |\zeta_{i+2}|$. Then,

$$f_{d-i} = \zeta_1 \ldots \zeta_{i-1} (\zeta_i + \zeta_{i+1} + \zeta_{i+2}) + \ldots \tag{14}$$

$$f_{d-i-1} = \zeta_1 \ldots \zeta_{i-1} (\zeta_i \zeta_{i+1} + + \zeta_i \zeta_{i+2} + \zeta_{i+1} \zeta_{i+2}) + \ldots \tag{15}$$

$$f_{d-i-2} = \zeta_1 \ldots \zeta_{i+2} + \ldots \ . \tag{16}$$

Hence, *on the limit*, we approximate:

$$\frac{f_{d-i-2}}{f_{d-i-1}} \simeq \frac{\zeta_i\zeta_{i+1}\zeta_{i+2}}{\zeta_i\zeta_{i+1} + +\zeta_i\zeta_{i+2} + \zeta_{i+1}\zeta_{i+2}} \tag{17}$$

$$\frac{f_{d-i-1}}{f_{d-i}} \simeq \frac{\zeta_i\zeta_{i+1} + +\zeta_i\zeta_{i+2} + \zeta_{i+1}\zeta_{i+2}}{\zeta_i + \zeta_{i+1} + \zeta_{i+2}} \tag{18}$$

$$\frac{f_{d-i}}{f_{d-i+1}} \simeq \zeta_i + \zeta_{i+1} + \zeta_{i+2} \ . \tag{19}$$

The moduli of those roots is therefore given by:

$$\sqrt[3]{\left|\frac{f_{d-i-2}}{f_{d-i-1}}\right|\left|\frac{f_{d-i-1}}{f_{d-i}}\right|\left|\frac{f_{d-i}}{f_{d-i+1}}\right|} \ . \tag{20}$$

The same formula extends for factors of any degree, provided they have roots in a circle and all other roots are far away from this circle.

It is useful to have a decision criterion for the existence of factors of degree greater than 1. We will do that for degree-2 factors, since this is the interesting case for real polynomials. See Ostrowskii [35] for more results.

Clearly, it is enough to consider the case of a polynomial f of degree 2:

$$f(x) = f_2x^2 + f_1x + f_0 = x^2 + (-\zeta_1 - \zeta_2)x + \zeta_1\zeta_2 \ . \tag{21}$$

In case $|\zeta_1| \gg |\zeta_2|$, we have:

$$-\log\left(\frac{|f_2|}{|f_1|}\right) + \log\left(\frac{|f_1|}{|f_0|}\right) \geq \log\frac{|\zeta_1|}{|\zeta_2|} \gg 0 \ . \tag{22}$$

In case $R = |\zeta_1| = |\zeta_2|$, we can bound:

$$-\log\left(\frac{|f_2|}{|f_1|}\right) + \log\left(\frac{|f_1|}{|f_0|}\right) < \log 4 \ . \tag{23}$$

However, we should look at the renormalized Newton diagram. In that case, we should divide equations (22) and (23) by 2^k , and expect that if $|\zeta_1| \neq |\zeta_2|$ then the following holds:

$$-2^{-k}\log\left(\frac{|f_2|}{|f_1|}\right) + 2^{-k}\log\left(\frac{|f_1|}{|f_0|}\right) \geq \sigma \tag{24}$$

where σ is an a priori bound on root moduli separation. It may be obtained from a probabilistic analysis (Lemma 6), or from any other a priori knowledge on the polynomial; for the choice of σ, notice that if $|\zeta_1| = |\zeta_2|$, ,

$$-2^{-k}\log\left(\frac{|f_2|}{|f_1|}\right) + 2^{-k}\log\left(\frac{|f_1|}{|f_0|}\right) < 2^{-k}\log 4 \ . \tag{25}$$

Thus, we may choose k such that $2^{-k}\log 4 < \sigma$. In case equation (24) is not satisfied, it is reasonable to assume that the two roots have the same modulus indeed.

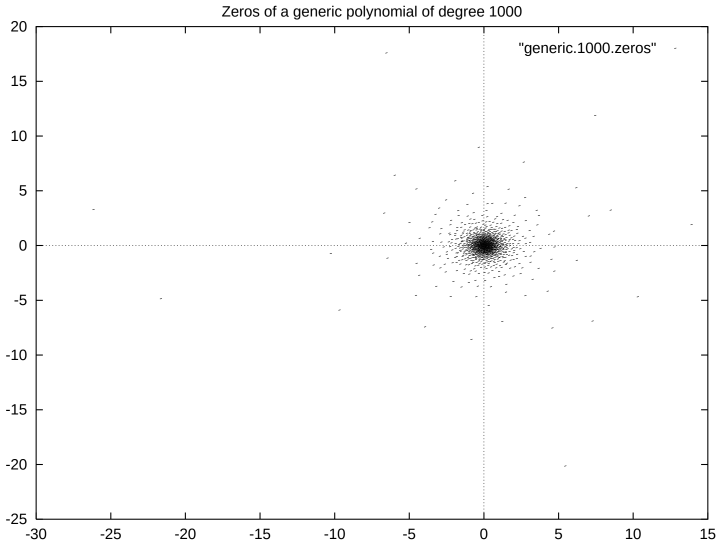

Figure 2: Zeros of a random polynomial

6 Numerical Results

The results discussed below are tentative, and our algorithm deserves further experimentation. Moreover, at this moment we are using a tentative algorithm to find the argument of the roots. Those matters will be dealt with in a subsequent paper [29]. The purpose of this section is to illustrate the numerical properties of Renormalized Graeffe Iteration, when applied to random real polynomials.

A C implementation of our algorithm was tested for pseudo-random real and complex polynomials. The arguments of the solutions were recovered using an algorithm derived from the Renormalized Graeffe Iteration.

IMPLEMENTATION DETAILS: Polynomials with zero coefficients do arise in practice, and we explain now how to deal with the 'log of zero' problem. The IEEE floating point arithmetic, implemented by most modern computers, has a few useful features for this sort of situation. When one asks a computer to produce $\log 0$, it returns a special IEEE value called $-\infty$. This is not an error, and further calculations can be carried out (as long as they are defined). If they are not defined (e.g. $\infty - \infty$) then some other special value, called a 'not a number' is returned.

This last case $(\infty - \infty)$ can appear during the computation of a renormalized sum. It can be dealt with testing a and b in Example 1 for finiteness. In case a or b is infinite, then c should be set to ∞. For an introduction of IEEE arithmetic, see [19] pages 45-48 or [12] pages 9-15. The correctness of the results was certified by estimates as in [27].

Experiments were performed in a Pentium 66 system running Linux operating system. Since the objective here was to illustrate the asymptotic behaviour of the algorithm, we did not perform experiments in other systems. Those would be necessary if one wanted to compare with other algorithms with same asymptotic properties. However, this goes far beyond the scope of this paper.

The table below shows the average and median user time in a Pentium- based computer, using 'double-extended' precision. Time does not include validation time. Ten pseudo-random polynomials were tested for each degree. The actual experimental data is plotted in Figure 6.

Degree	Real polynomials		Complex polynomials	
	avg time (s)	median time(s)	avg time (s)	median time (s)
100	0.87	0.87	1.19	1.18
200	3.23	3.24	4.35	4.30
300	7.07	7.07	9.99	9.73
400	12.60	12.58	17.28	17.10
500	19.41	19.38	26.75	26.71
600	27.67	27.73	37.02	35.96
700	37.50	37.32	51.30	49.91
800	48.89	48.72	65.39	63.61
900	61.56	61.06	79.28	78.74
1000	75.89	75.47	102.33	101.80

We also computed (approximately) the relative separation of the moduli of the solutions ζ_i:

$$\min_{|\zeta_i|<|\zeta_j|} \frac{|\zeta_i| - |\zeta_j|}{\sqrt{1+|\zeta_i|^2}} \ .$$

The values obtained are also plotted in figure 3

Further experimentation is necessary to obtain data about polynomials of degree $\gg 1000$. Indeed, due to underflow, we cannot represent random high-degree polynomials in the usual floating point representation.

7 Acknowledgments

JPZ was supported by CNPq and GM was partially supported by CNPq. The authors wish to thank the following people for their help: Jean-Pierre Dedieu, Carlos Isnard, Peter Kirrinnis, Welington de Melo, Victor Pan, Mike Shub, Jose Felipe da Silva, Steve Smale, Benar Fux Svaiter, Jean-Claude Yakoubsohn.

The authors gratefully acknowledge a large number of helpful comments by the anonymous referees, which helped to enhance the presentation tremendously.

References

[1] E. H. Bareiss. Resultant Procedure and the Mechanization of the Graeffe Process. *J. ACM* **7** pp 346-386, 1960.

[2] E. H. Bareiss. The numerical Solution of Polynomial Equations and the Resultant Procedures. In: A. Ralston and H. S. Wilf, eds, *Mathematical Methods for Digital Computers Vol II*, pp 185-214 John Wiley and Sons, Inc, New York, 1967.

[3] B. Beauzamy and J. Degot: Differential identities. *Transactions of the AMS* **347**, pp 2607-2619, 1995.

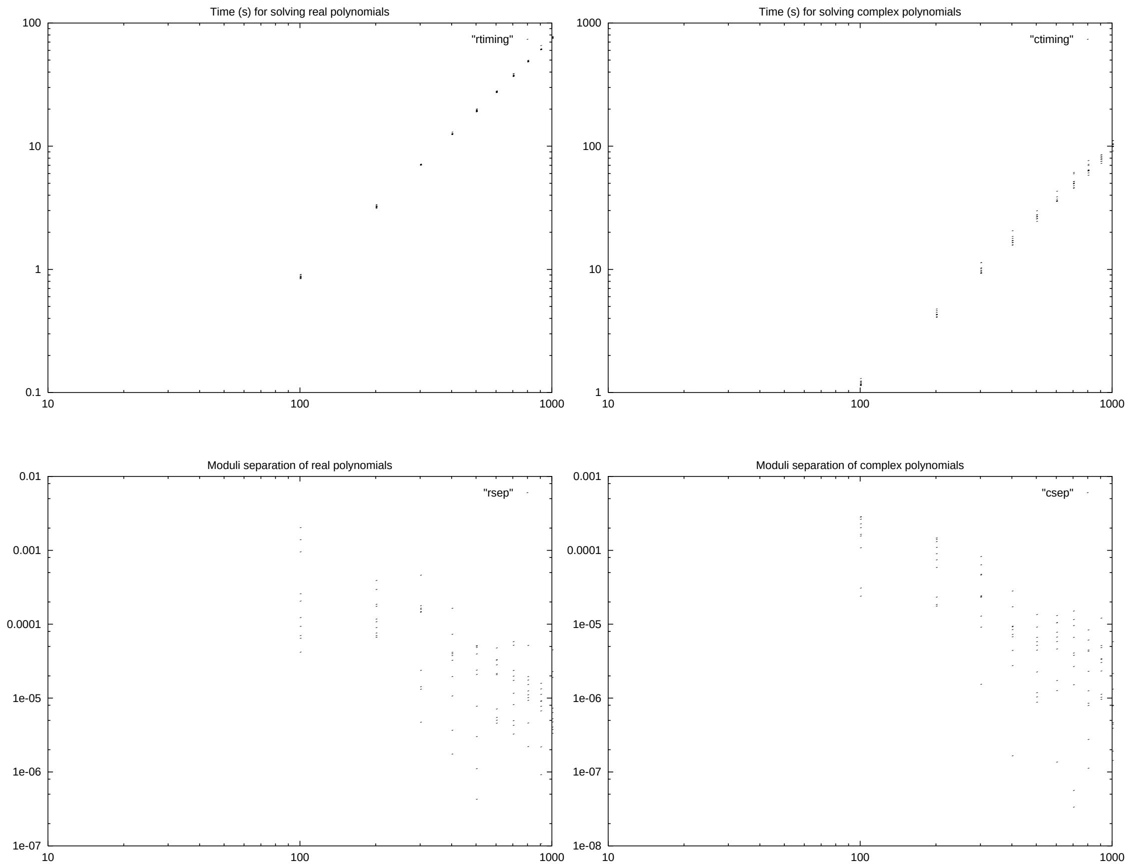

Figure 3: Time and separation of 100 random polynomials

[4] Dario Bini and Victor Pan. *Polynomial and Matrix Computations, Vol 1* Progress in Theoretical Computer Science, Birkhäuser, Boston, 1994.

[5] Dario Bini and Victor Pan, Graeffe's, Chebyshev-like and Cardinal's Processes for Splitting a Polynomial into Factors. *Journal of Complexity* **12**, pp 492-511, 1996.

[6] Sharon L. Blish and James H. Curry, On the Geometry of Factorization Algorithms. in: Eugene L. Allgower and Kurt Georg, eds, *Computational Solution of Nonlinear Systems of Equations.* Lectures in Applied Mathematics **26**, American Mathematical Society, Rhode Island, 1990.

[7] Lenore Blum, Felipe Cucker, Mike Shub and Steve Smale. *Complexity and Real Computation* Springer-Verlag, New York, 1998.

[8] Jean-Paul Cardinal, On Two Iterative Methods for Approximating the Roots of a Polynomial. *The Mathematics of Numerical Analysis. 1995 AMS-SIAM Summer Seminar in Applied Mathematics, July 17- August 11, 1995, Park City, UT.* Lectures in Applied Mathematics **32**, pp 165–188. American Mathematical Society, Rhode Island, 1996.

[9] Jean-Pierre Dedieu, A propos de la méthode de Graeffe. C.R. Acad. Sci. Paris, t. 309, pp 1019-1022, 1989.

[10] Jean-Pierre Dedieu, Approximate Solutions of Numerical Problems, Condition Number Analysis and Condition Number Theorem. in: James Renegar, Michael Shub, Steve Smale (eds), *The Mathematics of Numerical Analysis. 1995 AMS-SIAM Summer Seminar in Applied Mathematics, July 17- August 11, 1995, Park City, UT.* Lectures in Applied Mathematics **32**, pp 263-283. American Mathematical Society, Rhode Island, 1996.

[11] Jean-Pierre Dedieu, Condition Number Analysis for Sparse Polynomial Systems. in: Felipe Cucker and Mike Shub (eds), *Foundations of Computational Mathematics – Selected Papers of a Conference held at IMPA in Rio de Janeiro, Jan 1997*, pp 75–101. Springer-Verlag, Berlin, 1997.

[12] James W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[13] Alan Edelman and H. Murakami, Polynomial Roots from Companion Matrix Eigenvalues. *Mathematics of Computation* **64** No 210, pp 763-776, 1995.

[14] Ioannis Z. Emiris, André Galligo, and Henri Lombardi, Numerical Univariate Polynomial GCD. *The Mathematics of Numerical Analysis. 1995 AMS-SIAM Summer Seminar in Applied Mathematics, July 17- August 11, 1995, Park City, UT.* Lectures in Applied Mathematics **32**, pp 323-343. American Mathematical Society, Rhode Island, 1996.

[15] Mitchell J. Feigenbaum Quantitative Universality for a class of Nonlinear Transformations. *J. of Statistical Physics* **19**, no.1, 1978.

[16] S. Goedecker, Remark on Algorithms to Find Roots of Polynomials. *SIAM J. Sci. Computing* **15** No 5, pp 1059-1063, 1994.

[17] A. A. Grau. On the Reduction of Number Range in the Use of the Graeffe Process. *Journal of the ACM*, 1963.

[18] P. Henrici. *Applied and Computational Complex Analysis.* Wiley, New York, 1974.

[19] Nicolas J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[20] Alston S. Householder. Dandelin, Lobatchevskii or Graeffe ? *American Mathematical Monthly.*

[21] Anthony Hyman, *Charles Babbage, Pioneer of the Computer.* Princeton University Press, 1982.

[22] M.A.Jenkins and J.F.Traub. A Three-Stage variable shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. *Numer. Math.* **14**, 1970.

[23] Peter Kirrinnis, Partial Fraction Decomposition in $\mathbb{C}^n$ and simultaneous Newton Iteration for Factorization in $\mathbb{C}[z]$. *Journal of Complexity.* **14**:3, 378–444, 1998.

[24] Peter Kirrinnis, Newton Iteration Towards a Cluster of Polynomial Zeros. in: Felipe Cucker and Mike Shub (eds), *Foundations of Computational Mathematics – Selected Papers of a Conference held at IMPA in Rio de Janeiro, Jan 1997*, pp 193–215. Springer-Verlag, Berlin, 1997.

[25] Eric Kostlan *Random Polynomials and the Statistical Fundamental Theorem of Algebra.* Preprint, MSRI, 1987.

[26] Eric Kostlan, in *From Topology to Computation: Proceedings of the Smalefest (Berkeley, CA, 1990)*, 419–431, Springer, New York, 1993.

[27] Gregorio Malajovich, On generalized Newton algorithms: quadratic convergence, path-following and error analysis. Theoretical Computer Science **133**, 65-84 (1994).

[28] Gregorio Malajovich and Jorge P. Zubelli. A Fast and Stable Algorithm for Splitting Polynomials. *Computers and Mathematics with Applications*, **33**:3, 1–23, 1997.

[29] Gregorio Malajovich and Jorge P. Zubelli. Tangent Graeffe Iteration. Informes de Matemática. Serie B. Número 119, IMPA, Junho 1998 pp 1–34.

[30] Curtis T. McMullen. *Complex Dynamics and Renormalization* Annals of Mathematics Studies **135**. Princeton University Press, Princeton, New Jersey, 1994.

[31] Welington de Melo and S. van Strien, *One-Dimensional Dynamics.* Springer Verlag, 1993.

[32] Bernard Mourrain and Victor Y. Pan, Solving Special Polynomial Systems by Using Structured Matrices and Algebraic Residues. in: Felipe Cucker and Mike Shub (eds), *Foundations of Computational Mathematics – Selected Papers of a Conference held at IMPA in Rio de Janeiro, Jan 1997*, pp 287–304. Springer-Verlag, Berlin, 1997.

[33] C. Andrew Neff and John H. Reif. An Efficient Algorithm for the Complex Roots Problem. *Journal of Complexity* **12**, 81-115, 1996.

[34] Erich Novak and Henryk Woźniakowski, Topological Complexity of Zero-Finding. *Journal of Complexity* **12**, pp380-400, 1996.

[35] Alexandre Ostrowskii. Recherches sur la Méthode de Graeffe et les Zéros de Polynomes et des Séries de Laurent. *Acta Mathematica* **72**, 99-257, 1940.

[36] Jacob Palis and Floris Takens. *Hyperbolicity and Sensitive Chaotic Dynamics at Homoclinic Bifurcations* Cambridge Studies in Advanced Mathematics **35**. CUP, Cambridge, 1993.

[37] Victor Y.Pan, Deterministic Improvement of Complex Polynomial Factorization Based on the Properties of the Associated Resultant. *Computers Math. Applic.* **30** No 2, pp 71-94,1995.

[38] Victor Y.Pan, Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros. *Computers Math. Applic.* **31** No 12, pp 97–138, 1996.

[39] Victor Y.Pan. Solving a Polynomial Equation: Some History and Recent Progress. *SIAM Review* **39** No 1, 1997.

[40] Victor Y.Pan, Myongh-hi Kim, Akimou Sadikou, Xiaohan Huang and Aliong Zheng, On Isolation of Real and nearly Real Zeros of a Univariate Polynomial and its Splitting into Factors. *Journal of Complexity* **12**, pp 772–594, 1996.

[41] M.G. Pólya, Sur la méthode de Graeffe. *Comptes Rendus de l'Académie des Sciences de Paris,* 14 avril 1913.

[42] James Renegar, On the Cost of Approximating all Roots of a Complex Polynomial *Mathematical Programming* **32** (1985), 319-336.

[43] Michael Shub and Steve Smale, On the Complexity of Bezout's Theorem I - Geometric aspects. *Journal of the AMS*, **6**(2), (1993).

[44] Michael Shub and Steve Smale, On the complexity of Bezout's Theorem II - Volumes and Probabilities. in: F. Eysette and A. Galligo, eds: *Computational Algebraic Geometry.* Progress in Mathematics **109**, 267-285, Birkhauser (1993).

[45] Michael Shub and Steve Smale, Complexity of Bezout's Theorem III ; Condition number and packing. *Journal of Complexity* **9**, 4-14 (1993).

[46] Michael Shub and Steve Smale, Complexity of Bezout's Theorem IV ; Probability of success ; Extensions. Siam J. Numerical Analysis **33** No 1, pp 128-148, 1996.

[47] Michael Shub and Steve Smale, Complexity of Bezout's Theorem V: Polynomial time. *Theoretical Computer Science* **133**(1), 141-164 (1994)

[48] Steve Smale, Complexity Theory and Numerical Analysis. Preprint, Hong Kong, 1996. To appear, *Acta Numerica.*

[49] Arnold Schönhage, *The Fundamental Theorem of Algebra in Terms of Computational Complexity – Preliminary Report* Preprint, Tübingen, 1982.

[50] Arnold Schönhage, Equation solving in Terms of Computational Complexity. *Proceedings of the International Congress of Mathematicians, Berkeley, 1986.* 131-153, American Mathematical Society, 1987

[51] Kim-Chuan Toh and Lloyd N. Trefethen Pseudozeros of Polynomials and Pseudospectra of companion matrices. *Numer. Math.* **68**, pp 403–425 (1994).

[52] J.V.Uspensky. *Theory of equations.* McGraw Hill, New York, 1948.

[53] Victor Vassiliev Topological Complexity of Root-Finding Algorithms. *The Mathematics of Numerical Analysis. 1995 AMS-SIAM Summer Seminar in Applied Mathematics, July 17- August 11, 1995, Park City, UT.* Lectures in Applied Mathematics **32**, pp 831–856. American Mathematical Society, Rhode Island, 1996.

[54] Hermann Weyl. *The Theory of Groups and Quantum Mechanics* Dover Publications, Inc. c.1931, 1951.