Tangent Graeffe Iteration

Gregorio Malajovich and Jorge P. Zubelli

May 29, 1998

Abstract

Graeffe iteration was the choice algorithm for solving univariate polynomials in the XIX-th and early XX-th century. In this paper, a new variation of Graeffe iteration is given, suitable to IEEE floating-point arithmetics of modern digital computers.

We prove that under a certain generic assumption the proposed algorithm converges. We also estimate the error after N iterations and the running cost.

The main ideas from which this algorithm is built are: classical Graeffe iteration and Newton Diagrams, changes of scale (renormalization), and replacement of a difference technique by a differentiation one.

The algorithm was implemented successfully and a number of numerical experiments are displayed.

Contents

List of Algorithms

1 Introduction

Many present day numerical algorithms have originated in highly acclaimed methods dating from last century or even earlier. Such is certainly the case of Euler's method or of Newton's method, whose numerical and theoretical consequences still impact us today [13, 31, 32, 33, 35, 34, 37].

Graeffe's classical method for finding simultaneously all roots of a polynomial was introduced independently by Graeffe, Dandelin and Lobatchevsky [9]. Its simplicity, as well as importance throughout last century indicate its potential as an effective numerical algorithm.

Surprisingly, Graeffe's method has not received much attention in present day numerical computations. Very few modern discussions about it or its applications can be found. See the review by V. Pan [26], and also [2, 3, 4, 6, 14, 19, 20, 22, 25, 27, 30].

One of the main reasons for Graeffe's lack of popularity stems from the fact that its traditional form leads to exponents that easily exceed the maximum allowed by floating-point arithmetic. Other reasons, such as the "chaotic" behavior of the arguments of the roots of the iterates contribute to such stigma.

In this work we present a version of Graeffe's algorithm, which is well suited for floating-point arithmetic computations. Furthermore, it has excellent complexity and memory allocation characteristics. Our method computes both the moduli and the argument of all the roots, provided that certain generic conditions are satisfied. These claims are backed by our theoretical results presented in the next section, and proved throughout the paper, as well as the numerical experiments presented in the end of the paper.

The main ingredients in our approach are the following:

- The idea of renormalizing the relevant operations at each iteration step, akin to what is done in dynamical systems and physics [17, 21].
- The idea of using the differential of our Renormalized Graeffe iteration as a way of keeping the information concerning the argument of the roots.
- A renormalized version of Newton's diagram that allows us to recognize and locate pairs of conjugate roots, as well as roots of higher multiplicity.

The first idea mentioned above was developed in our earlier work [20], which in a certain sense laid the conceptual framework for our present approach. It is *not* however essential in understanding the proofs presented herein.

The second ingredient mentioned above is explained and motivated in Section 4. In rather vague terms it could be compared to the advantage of using derivatives, when those are available, as compared to using differences.

Finally, the concept of Newton's diagram, as well as the power of Graeffe's method was present throughout Ostrowski's masterpiece [23]. While writing the present paper we could not help but wonder what would have been the outcome of that research if he had available at that point the present day technology of high speed computers.

1.1 Main Result

We will introduce an algorithm for solving real and complex univariate polynomials. The following genericity condition will be required at input:

Definition 1. A *real* polynomial f will be called *circle free* if, and only if, for any couple ζ, ξ of distinct roots of f, one has either $|\zeta| \neq |\xi|$, or $\zeta = \bar{\xi}$.

Definition 2. A *complex* polynomial f will be called *circle free* if, and only if, for any couple ζ, ξ of distinct roots of f one has $|\zeta| \neq |\xi|$.

It is obvious that given any real polynomial f, one can obtain a circle free polynomial by (pre)composing it with a conformal transform of the form:

$$\begin{array}{rccc} \varphi : & \bar{\mathbb{C}} & \rightarrow & \bar{\mathbb{C}} \\ & x & \mapsto & \varphi(x) = \frac{x \cos\theta - \sin\theta}{x \sin\theta + \cos\theta} \end{array}$$

and then clearing denominators; for all but a finite number of $\theta \in (-\pi, \pi]$, the resulting polynomial

$$\tilde{f}(x) = (x \sin\theta + \cos\theta)^d f\left(\frac{x\cos\theta - \sin\theta}{x \sin\theta + \cos\theta}\right)$$

is circle free.

Tangent Graeffe Iteration will be shown to converge for all circle free polynomials; Given an arbitrary polynomial, one can first find all zero roots (in the obvious way), apply a random conformal transform, then Tangent Graeffe Iterations, and finally recover the roots of the original polynomial.

When counted with multiplicity, the roots of a circle free polynomial can be canonically ordered by:

1. $|\zeta_1| \leq \cdots \leq |\zeta_d|$
2. In the real case,
   - 2.1. If $|\zeta_i| = |\zeta_{i+1}|$ then $\zeta_i = \overline{\zeta_{i+1}}$.
   - 2.2. If $i = 1$ or $|\zeta_{i-1}| < |\zeta_i|$ then Im $\zeta_i \geq 0$.

If we assume that all the arithmetical operations are performed exactly (including transcendental), the mathematical properties of the algorithm can be summarized by:

Theorem 1. *Let f be a real (resp. complex) circle free degree d polynomial, not vanishing at 0. Denote by ζ the vector of all the roots of f with multiplicity canonically ordered as above.*

Then, a total of N iterations of Renormalized Tangent Graeffe (Algorithm 6) produces $\zeta^{(N)} \in \mathbb{C}^d$, such that

$$\zeta^{(N)} \longrightarrow \zeta \qquad (\text{as } N \rightarrow \infty).$$

The running time for each iteration is $O(d^2)$ exact arithmetic operations (including transcendental operations). The relative error bound in each coordinate after N iterations is $2^{-2^{N-C}}$, where C depends on f.

1.2 What the Graeffe Iteration is; Its historical weaknesses

In this section we shall briefly review the main ideas behind the method and describe also some of its weaknesses.

Graeffe iteration maps a degree d polynomial $f(x)$ into the degree d polynomial

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

If $\zeta_1, \zeta_2, \ldots \zeta_d$ are the roots of f, then the roots of Gf are $\zeta_1^2, \zeta_2^2, \ldots \zeta_d^2$

Assume that $g = G^N f$ is the N-th iterate of f. Then, assuming that f is monic, the coefficients of $g(x) = g_0 + g_1 x + \cdots + g_d x^d$ satisfy:

$$\begin{aligned}
g_0 &= (-1)^d \ \sigma_d \ \left(\zeta_1^{2^N}, \zeta_2^{2^N}, \cdots, \zeta_d^{2^N}\right) \\
g_1 &= (-1)^{d-1} \ \sigma_{d-1} \ \left(\zeta_1^{2^N}, \zeta_2^{2^N}, \cdots, \zeta_d^{2^N}\right) \\
&\vdots \\
g_j &= (-1)^{d-j} \ \sigma_{d-j} \ \left(\zeta_1^{2^N}, \zeta_2^{2^N}, \cdots, \zeta_d^{2^N}\right) \\
&\vdots \\
g_d &= \sigma_0 = 1
\end{aligned}$$

where σ_j is the j-th elementary symmetric function. In the particular case that $|\zeta_1| < |\zeta_2| < \cdots < |\zeta_d|$, we can further approximate

$$\begin{aligned}
g_0 &= (-1)^d \zeta_1^{2^N} \zeta_2^{2^N} \ldots \zeta_d^{2^N} \\
g_1 &\simeq (-1)^{d-1} \zeta_2^{2^N} \ldots \zeta_d^{2^N} \\
&\vdots \\
g_j &\simeq (-1)^{d-j} \zeta_{j+1}^{2^N} \ldots \zeta_d^{2^N} \\
&\vdots \\
g_d &= \sigma_0 = 1
\end{aligned}$$

Hence, it is possible to determine

$$\zeta_j^{2^N} \simeq -\frac{g_j}{g_{j+1}} \ .$$

We stress two main weaknesses. The first big weakness of classical Graeffe iteration is coefficient growth. As the coefficients of g_j grow doubly exponentially in the number of iterations, the exponent (not the mantissa) of the floating-point system gets overflowed:

Example 1. Let f have roots 1, 2, 3, 4. Then the N-th Graeffe iterate of f has roots 1, 2^{2^N}, 3^{2^N}, 4^{2^N}. The coefficient g_0 is 24^{2^N}. If $N = 8$, then g_0

is approximately 1.68×2^{1173}, while IEEE double precision numbers used in most modern computers cannot contain floating point values more than 2^{1024}, since the exponent is represented by 11 bits (sign included) [8]. (As a matter of fact, the representation is a little more complicated, as it allows for 'subnormal' numbers [5]). Therefore, we would have an overflow when computing the 8-th Graeffe iterate of f.

Example 2. On the example above, assume that f would have an additional root 1.01. Namely,

$$\begin{aligned} f(x) &= (x-1)(x-1.01)(x-2)(x-3)(x-4) \\ &= -24.24 + 74.5x - 85.35x^2 + 45.1x^3 - 11.01x^4 + x^5 \end{aligned}$$

We will show that 8 Graeffe iterations are not enough to compute the first root (namely 1) to an accuracy of 10^{-4}. However, as shown in Example 1, 8 iterates are enough to overflow the IEEE double precision number system.

Indeed, the first root is obtained as:

$$\begin{aligned} \zeta^{2^N} \simeq -\tfrac{g_0}{g_1} &= -\tfrac{1.01^{2^N} 24^{2^N}}{(1^{2^N}+1.01^{2^N})24^{2^N}} \\ &= \tfrac{1}{1+1.01^{-2^N}} \\ &\simeq 0.927 \quad (\text{ for } N=8). \end{aligned}$$

Thus,

$$\zeta \simeq 1 - 2.9 \times 10^{-4}$$

The error obtained is therefore larger than 10^{-4}.

The introduction of the idea of renormalization allows us to avoid coefficient growth, and replace a diverging algorithm by a convergent one (See Section 2 and also [20]). Alternative approaches for the number range growth are suggested in [7] and in [6].

A certain geometrical invariant of the polynomial, the *limiting Newton diagram*, appears naturally in the context of Renormalized Graeffe Iteration. It allows to recover the information about multiple roots and pairs of roots. (See [23]).

In Section 3, we give a procedure to obtain the limiting Newton diagram of a given polynomial. It is effective in the sense that, if we can bound the separation

$$\max_{|\zeta_i|>|\zeta_j|} \frac{|\zeta_i|}{|\zeta_j|} ,$$

then we can effectively identify the multiple roots and pairs of roots. It will converge, and eventually provide the list of multiple roots and pairs for any circle-free polynomial, in a finite (but unknown, not effective) number of iterations.

The other big weakness of classical Graeffe iteration is the fact that it returns the moduli of the roots, but not the actual roots. As a matter of fact, information about the argument of the roots is lost, and should be recovered by other means:

Example 3. Consider polynomials $f(x) = x^2 - 2x + 1$, $g(x) = x^2 - 1$, $h(x) = x^2 + 1$. After two Graeffe iterations, all the three polynomials are mapped into $f(x)$.

Many algorithms have been proposed to recover the arguments (See [26]). In this paper, we will differentiate the Graeffe iteration operator, and obtain an iteration defined on the appropriate tangent bundle. This new operator will define a mapping between 1-jets of polynomials. By the latter we mean expressions of the form $f(x) + \epsilon \dot{f}(x)$, where ϵ is a formal parameter. This procedure is discussed in section 4.

In section 5, we compare the numerical behavior of Renormalized Tangent Graeffe Iteration to other publicly available algorithms.

2 Renormalizing Graeffe

2.1 The Renormalized Graeffe Iteration

Example 2 shows a typical behavior of classical Graeffe iteration performed by digital computers [7]. In order to avoid that sort of overflow, the authors introduced in [20] the *Renormalized Graeffe Iteration.* Although the details and the mathematical foundations of the algorithm are described in [20], to keep the present work self-contained, we give below a very short description of the main ideas:

One should consider the computation of $G^N f$ as divided in several *renormalization levels.*

$$\begin{array}{ll} \text{Level 0} & \text{Coefficients of } f \\ \text{Level 1} & \text{Coefficients of } Gf \\ \text{Level 2} & \text{Coefficients of } G^2 f \\ & \vdots \\ \text{Level N} & \text{Coefficients of } G^N f \end{array}$$

At renormalization level N, all coefficients g_j^N of $G^N f$ should be represented in coordinates

$$r_j^{(N)} = -2^{-N} \log |g_j^N| \ ,$$

and

$$\alpha_j^{(N)} = g_j^N / |g_j^N| \ .$$

Therefore, we shall obtain convergence of the radial coordinates r_j^N (at least in the case of roots of different moduli). The dynamics of the angular coordinates α_j^N is typically chaotic.

In order to pass from level N to level $N+1$, a *Renormalized Graeffe Operator* was defined in [20]. Intermediate computations were performed in coordinates r_j^N and α_j^N by means of renormalized arithmetic operations. For instance, the renormalized sum (r, α) of (r_1, α_1) and (r_2, α_2) can be defined (in renormalization level N) by

$$r = -2^{-N} \log |\alpha_1 e^{-2^N r_1} + \alpha_2 e^{-2^N r_2}|$$

$$\alpha = \frac{\alpha_1 e^{-2^N r_1} + \alpha_2 e^{-2^N r_2}}{|\alpha_1 e^{-2^N r_1} + \alpha_2 e^{-2^N r_2}|}$$

Renormalized sum can be computed without overflow by the formula in Algorithm 1. This is a simplified, non-optimal version of renormalized sum. Notice that one or two of the inputs can be the renormalization of 0, i.e., ∞. Under the usual conventions, ∞ is greater than any real number. Therefore, if only one of the arguments is ∞, the correct result will be returned.

Algorithm 1 RenSum (r_1, α_1, r_2, α_2, p)

{ It is assumed that r_1 and r_2 are real numbers or $+\infty$, and that $|\alpha_1| = |\alpha_2| = 1$. The number p should be equal to 2^N, where N is the renormalization level. This routine computes (in renormalized coordinates !) the sum of $\alpha_1 e^{-pr_1}$ and $\alpha_2 e^{-pr_2}$. }
if $r_1 = r_2 = +\infty$ **then**
  return $+\infty, 1$
end if
$\Delta \leftarrow r_2 - r_1$
if $\Delta \geq 0$ **then**
  $t \leftarrow \alpha_1 + \alpha_2 e^{-p\Delta}$
  return $r_1 - \frac{\log(|t|)}{p}, t/|t|$
else
  $t \leftarrow \alpha_2 - \alpha_1 e^{p\Delta}$
  return $r_2 - \frac{\log(|t|)}{p}, t/|t|$
end if

A few extra mathematical ideas related to the renormalized Graeffe operators, as well as some other mathematical results can be found in [20].

2.2 The Renormalized Newton Diagram

The first goal of this section is to introduce the concept of Renormalized Newton diagram, which is going to play a key role in the practical implementation of the algorithm discussed in this paper. The second is to prove a convergence result based on such idea using some earlier results of Ostrowski's.

We start by reviewing the concept of Newton diagram, which has been used extensively by Ostrowski, Puiseux and Dumas, among others.

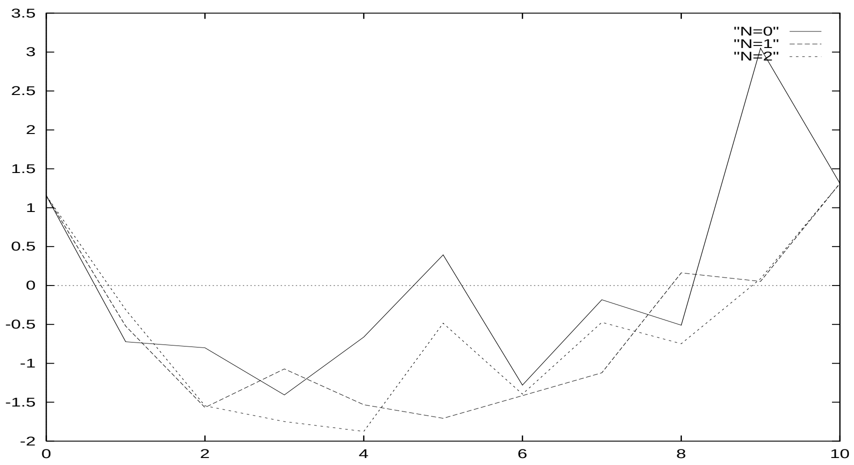

Figure 1: The function $r^{(N)}(i)$, for $N = 0, 1, 2$

Let

$$f = \sum_{i=0}^{d} f_i x^i \ ,$$

be a degree d polynomial. As before, we denote by $G^N f$ the N-th Graeffe iterate of f.

We order the roots of f in nondecreasing order of their moduli, to wit:

$$|\zeta_1| \leq \cdots \leq |\zeta_d| \ . \tag{1}$$

If the above inequalities are all strict, then

$$\lim_{N\to\infty} 2^{-N} \log \left(\frac{|(G^N f)_i|}{|(G^N f)_{i+1}|} \right) = \log(|\zeta_{i+1}|) \ .$$

For each N, consider the piecewise linear function $r^{(N)} : [0, d] \to \mathbb{R} \cup \{+\infty\}$ satisfying

$$r^{(N)}(i) \stackrel{\text{def}}{=} -2^{-N} \log |(G^N f)_i| \ .$$

Notice that under the above assumptions, $r^{(N)}$ is convex for sufficiently large N. (See figure 1). Indeed, since $|\zeta_{i+1}| > |\zeta_i|$, the inclinations satisfy (for large N)

$$r^{(N)}(i+1) - r^{(N)}(i) \ \geq \ r^{(N)}(i) - r^{(N)}(i-1) \ .$$

It is easy to see that if two consecutive roots, say ζ_i and ζ_{i+1}, have approximately the same absolute value, then the three corresponding points

$$\left(i-1, r^{(N)}(i-1)\right), \left(i, r^{(N)}(i)\right), \left(i+1, r^{(N)}(i+1)\right)$$

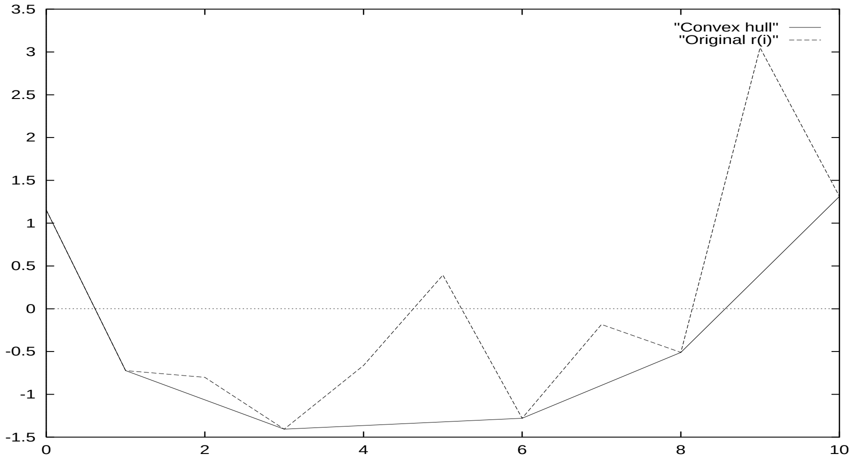

Figure 2: The function $r^{(N)}(i)$ and its Convex Hull

will be approximately aligned. Furthermore, the functions $r^{(N)}$ converge to a piecewise linear convex function.

However, if the inequalities in (1) are not strict, the functions $r^{(N)}$ may fail to converge.

Example 4. Let $f(x) = (x-1)(x-e^{i\theta})$. Then its N-th Graeffe iterate is

$$f^{(N)}(x) = x^2 - (1 + e^{2^N i\theta})x + e^{2^N i\theta}$$

Therefore, we have $r^{(N)}(0) = r^{(N)}(2) = 0$, but we also have

$$r^{(N)}(1) = -2^{-N} \log |1 + e^{2^N i\theta}| = -2^{-N} \log |2 \cos 2^{N-1}\theta| \quad .$$

Depending on the choice of θ, this last value can range anywhere from $-2^{-N} \log 2$ to $+\infty$.

This is one of the reasons for introducing the convex hull of each $r^{(N)}$, that will be subsequently called the *Renormalized Newton Diagram.* (See figure 2).

Our approach has the advantage of simplifying some of the arguments by Ostrowski in [23] by providing plain convergence of Renormalized Newton Diagrams; however, we will quote several of the results by Ostrowski in the sequel.

One of the major goals of Ostrowski in [23] was to obtain effective bounds for the moduli of roots. This was possible by introducing of the *majorant* of a given polynomial:

Definition 3. A majorant of a given polynomial is any other polynomial, of same degree, with nonnegative coefficients greater than or equal to the given polynomial's coefficients.

The first step of Ostrowski ' s construction is Newton's majorant:

Definition 4. A polynomial $A = \sum_{i=0}^{d} a_i x^i$ with nonnegative coefficients is called *normal* if the following conditions hold:

1. If $a_i > 0$ and $a_j > 0$ for $i > j$, then $a_l > 0$ for all $i < l < j$.
2. For $l = 1, \ldots, d-1$,

$$a_l^2 \geq a_{l-1} a_{l+1} \ .$$

A normal majorant T of f is called *minimal* if for any other majorant T' of f we have

$$T_j \leq T'_j \ , \ j = 0, 1, \ldots, d. \tag{2}$$

Notice that Condition 2 above means that the graph of the points of the form $(l, -\log(r_l))$, for $l = 0, \ldots, d$ is convex.

In the language of majorants,

Proposition 1 (Ostrowski[23]). *Any polynomial*

$$f = \sum_{j=0}^{d} f_j z^j$$

possesses a unique minimal normal majorant,

$$\mathcal{M}_f = \sum_{j=0}^{d} T_j z^j \ .$$

The result can be proved by using the convex hull ϕ of the function $-\log|f_i|$. and constructing the polynomial $\mathcal{M}_f$ as the polynomial with positive coefficients $(\mathcal{M}_f)_j = e^{-\phi(j)}$. We refer the interested reader to Ostrowski's work [23, 24].

We remark that if the polynomial has roots of strictly increasing moduli, then the coefficients $T_i^{(N)}$ of the Newton's majorant of $G^N f$ coincide with $|f_i^{(N)}|$ for N sufficiently large and $i = 0, 1, \ldots, d$.

However, the introduction of Newton Diagram allows us to consider the general situation of possibly many roots of same moduli. As before, we order them in nondecreasing order and consider the indices

$$i_0 = 0 < i_1 < i_2 < \cdots < i_l < i_{l+1} = d$$

as 0, d and exactly those integers i between 1 and $d-1$ such that $|\zeta_i| < |\zeta_{i+1}|$. This way, we have that

$$|\zeta_{i_{j-1}}| < |\zeta_{i_{j-1}+1}| = \cdots = |\zeta_{i_j}| < |\zeta_{i_j+1}| \ .$$

The fundamental result, in this case is

$$\lim_{N\to\infty} 2^{-N} \log \frac{|f_{i_j}^{(N)}|}{|f_{i_{j+1}}^{(N)}|} = (i_{j+1} - i_j) \log|\zeta_i| ,$$

for $i_j < i \le i_{j+1}$ and $0 \le j \le l$. (c.f. equation (79.8) of [24]). In the language of Renormalized Newton Diagrams, that very same equation can be written as:

$$\log|\zeta_i| = \lim_{N\to\infty} \frac{r^{(N)}(i_{j+1}) - r^{(N)}(i_j)}{i_{j+1} - i_j} , \qquad \text{for } i_j < i \le i_{j+1}.$$

As remarked by Ostrowski, the above formulae are only useful in the determination of the moduli of the roots if we know "a priori" the values $i_1 < i_2 < \cdots < i_l$. This is obviously not the case in most applications. Instead, Ostrowski's results provide effective bounds for the convergence of the $r^{(N)}$, and thus for the values of the $|\zeta_i|$' s.

Theorem 2 (Ostrowski[23],Theorem IX.3). *Let*

$$\varrho(\nu) \stackrel{def}{=} 1 - 2^{-1/\nu} ,$$

and

$$R_\nu^{(N)} \stackrel{def}{=} \frac{T_{\nu-1}^{(N)}}{T_\nu^{(N)}} ,$$

then

$$\varrho(\nu) < \frac{|\zeta_\nu|^{2^N}}{R_\nu^{(N)}} < \frac{1}{\varrho(d-\nu+1)} \qquad \nu = 1, \ldots, d. \tag{3}$$

As a consequence of the above estimate, Ostrowski gets the following bound

$$(2d)^{-2^{-N}} < \frac{|\zeta_\nu|}{(R_\nu^{(N)})^{2^{-N}}} < (2d)^{2^{-N}}$$

Corollary 1. *If $r^{(N)}(i)$ denotes the i-th ordinate of the N-th Renormalized Newton Diagram, then*

$$\lim_{N\to\infty} \left(r^{(N)}(i) - r^{(N)}(i-1) \right) = \log|\zeta_i|$$

for $i = 1, \ldots, d$. Furthermore, the error is bounded from above by

$$2^{-N} \log(2d) .$$

This is indeed a strong result, since nothing is assumed on the coefficients or the roots of the original polynomial. However, it is possible to get a better error bound by assuming a minimal separation on the moduli:

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

We note that in the above formula, if ϵ is well defined (i.e. there are at least two roots of different modulus) then it is non-zero.

For generic configurations, a lower bound on the separation of root moduli can be expected. One reason for this is Lemma 7 from [20]. Namely,

Lemma 1. *Let f be a randomly distributed polynomial w.r.t the unitary invariant distribution and denote its roots by $\{\zeta_i\}_{i=1}^d$. The probability*

$$p = \mathrm{Prob}\left[\min_{|\zeta_i|>|\zeta_j|} \frac{|\zeta_i|}{|\zeta_j|} > 1 + \epsilon\right]$$

satisfies

$$p > 1 - \frac{d(d-1)\epsilon(1+\epsilon)}{6} > 1 - \frac{1}{3}d(d-1)\epsilon \ .$$

The proof is omitted here, since the results of this paper do not depend on it.

Hence, if we set

$$\delta \stackrel{\text{def}}{=} \frac{1}{3}d(d-1)\epsilon \ ,$$

we have that with probability larger than $1-\delta$, the modulus of separation

$$\rho \stackrel{\text{def}}{=} \min_{|\zeta_i|>|\zeta_j|} \frac{|\zeta_i|}{|\zeta_j|} > 1 + \frac{3\delta}{d(d-1)} = 1 + \epsilon \ .$$

Those estimates can be used to bound the average number of Graeffe iterations needed to solve a polynomial equation within a given accuracy. In other words, to bound the probabilistic convergence speed of the algorithm on a random configuration of zeros.

3 Computation of the Newton Diagram

3.1 Algorithm and Main Statements

The main issue in this section is the following:

We are given a certain polynomial g, obtained after a few Graeffe iterations of a polynomial f. The roots of g are $Z_1, \ldots, Z_d$, and we order them so that:

$$|Z_1| \le |Z_2| \le \cdots \le |Z_d|$$

We want to know which of the inequalities are strict. We do not know the actual value of the Z_i's, we know only the coefficients of g, i.e., the symmetric functions of the Z_i's.

We are also ready to assume that

$$R \stackrel{\text{def}}{=} \min_{|Z_{i+1}|>|Z_i|} \frac{|Z_{i+1}|}{|Z_i|}$$

is a large real number. Indeed, if $\zeta_1, \ldots, \zeta_d$ are the roots of f, always ordered such as $|\zeta_1| \leq \cdots \leq |\zeta_d|$, then

$$\rho \stackrel{\text{def}}{=} \min_{|\zeta_{i+1}|>|\zeta_i|} \frac{|\zeta_{i+1}|}{|\zeta_i|}$$

is always strictly greater than one. Hence, given any $A > 0$, by performing $N \geq \log_2 \frac{\log A}{\log \rho}$ iterations, we can assume that $R = \rho^{2^N} \geq A$.

Recall that the Renormalized Newton Diagram of g is the convex hull of the function $i \mapsto -2^{-N} \log g_i$. As N grows, the Renormalized Newton Diagram of g converges to the convex hull of

$$i \mapsto -\log|f_0| + \sum_{j \leq i} \log|\zeta_i| \ .$$

However, we want to be able to decide in *finite time* what are the sharp corners of the convex hull of $i \mapsto \log|f_d| + \sum_{j\leq i} \log|\zeta_j|)$. As in the preceding section, we write:

$$r_i = -2^{-N} \log|g_i|$$

where g is the N-th iterate of f. Notice that we dropped the superscript N of $r_i^{(N)}$.

Proposition 2. *Let $g = G^N f$, where f is a degree d polynomial and G denotes the Graeffe iteration. Let Z, ζ and ρ be as above. Assume that*

$$N > 3 + \log_2 \frac{d \log 2}{\log \rho} \ .$$

Then, Algorithm 2 below with input N, d, r, ρ produces the list of the sharp edges of the convex hull of $i \mapsto \sum_{j\leq i} \log|\zeta_i|)$.

Remark: Algorithm 2 has running time $O(d)$.

3.2 Some Estimates about Symmetric Functions

In order to prove Proposition 2, we need a few estimates about symmetric functions. First of all, let $I = \{i : |Z_i| < |Z_{i+1}|\} \cup \{0, d\}$ be the set of sharp

Algorithm 2 Strict Convex Hull (N, d, r, ρ)

{ Create a list Λ, containing initially the element $\Lambda_0 = 0$}
$j \leftarrow 0$;
$\Lambda_j \leftarrow 0$;

{The error bound below will follow from Lemma 5.}
$R \leftarrow \rho^{2^N}$
$E \leftarrow \frac{1}{2}\left(2^{-N+1}\log(2^d + 2^d R^{-1}) - 2^{-N+2}\log(1 - 2^d R^{-1}) + \frac{\log\rho}{2}\right)$

{Now, we will try to add more points to the list Λ. At each step, we want to ensure that we have always a convex set.}

for $i \leftarrow 1$ to d **do**

{We discard all the points in Λ that are external to the convex hull of Λ and the new point. Let Λ_j be the last element of Λ}

while $j > 0$ and $\frac{r_{\Lambda_j} - r_{\Lambda_{j-1}}}{\Lambda_j - \Lambda_{j-1}} > \frac{r_i - r_{\Lambda_j}}{i - \Lambda_j} - E$ **do**
$j \leftarrow j - 1$
end while

{Now, we append the point i}
$j \leftarrow j + 1$
$\Lambda_j \leftarrow i$;
end for

Return $(\Lambda_0, \cdots, \Lambda_j)$

corners of the limiting Renormalized Newton Diagram. As before, let σ_k denote the k-th elementary symmetric function,

$$\sigma_k(Z) = \sum_{j_1 < \cdots < j_k} Z_{j_1} \ldots Z_{j_k}$$

Then,

Lemma 2. *For $i \in I$ we have*

$$\sigma_{d-i}(Z) = Z_{i+1} Z_{i+2} \ldots Z_d (1 + c) \ ,$$

where $|c| \leq \left(\binom{d}{i} - 1\right) R^{-1} \leq 2^d R^{-1}$

Proof of Lemma 2. Write

$$\sigma_{d-i}(Z) = \sum_{j_1 < \cdots < j_{d-i}} Z_{j_1} \ldots Z_{j_{d-i}}$$

In the sum above, $|Z_{j_1} \dots Z_{j_{d-i}}| \leq R^{-1}|Z_{i+1}Z_{i+2}\dots Z_d|$ for any choice of j_1, $\dots$ j_{d-i} except $i+1, \dots d$. Since there are $\binom{d}{i} - 1$ other terms, we obtain that:

$$|\sigma_{d-i}(Z) - Z_{i+1}Z_{i+2}\dots Z_d| \leq \left(\binom{d}{i} - 1\right) R^{-1}|Z_{i+1}Z_{i+2}\dots Z_d|$$

□

Definition 5. We will say that i_1 and i_2 are *successive elements* of I if and only if:

1. $i_1 \in I$
2. $i_1 < i < i_2 \Rightarrow i \notin I$
3. $i_2 \in I$

Lemma 3. *Let i_1 and i_2 be successive elements of I, and let $i_1 < l < i_2$. Then*

$$|\sigma_{d-l}(Z)| \leq \left(\binom{i_2 - i_1}{i_2 - l} + c'\right) |Z_{i_2}|^{i_2 - l}|Z_{i_2+1}|\dots|Z_d|$$

with $c' \leq \left(\binom{d}{i} - \binom{i_2-i_1}{i_2-l}\right) R^{-1} < 2^d R^{-1}$

Proof of Lemma 3. Write

$$\begin{aligned} \sigma_{d-l}(Z) &= \sum_{j_1 < \dots < j_{d-l}} Z_{j_1}\dots Z_{j_{d-l}} \\ &= \sum{}' Z_{j_1}\dots Z_{j_{d-l}} + \sum{}'' Z_{j_1}\dots Z_{j_{d-l}} \end{aligned}$$

where $\sum'$ ranges over the j such that $i_1 < j_1 < \dots < j_{i_2-l} < i_2 + 1$ and $j_{i_2-l+1} = i_2+1, \dots, j_{d-l} = d$. Of course, $\sum''$ ranges over all the other terms.

We can rewrite $\sum'$ as:

$$\sum{}' = Z_{i_2+1}Z_{i_2+2}\dots Z_d \left(\sum_{i_1 < j_1 < \dots < j_{i_2-l} \leq i_2} Z_{j_1}\dots Z_{j_{i_2-l}}\right)$$

Hence,

$$|\sum{}'| \leq \binom{i_2 - i_1}{i_2 - l}|Z_{i_2}|^{i_2-l}|Z_{i_2+1}|\dots|Z_d|$$

The terms in $\sum''$ are all smaller than $R^{-1}|Z_{i_2}|^{i_2-l}|Z_{i_2+1}|\dots|Z_d|$. Since there are $\binom{d}{i} - \binom{i_2-i_1}{i_2-l}$ of them,

$$|\sum{}''| < \left(\binom{d}{i} - \binom{i_2 - i_1}{i_2 - l}\right) R^{-1}|Z_{i_2}|^{i_2-l}|Z_{i_2+1}|\dots|Z_d|$$

Adding those two bounds, we obtain indeed:

$$|\sigma_{d-l}(Z)| \le \left(\binom{i_2-i_1}{i_2-l} + \left(\binom{d}{i} - \binom{i_2-i_1}{i_2-l}\right) R^{-1}\right) |Z_{i_2}|^{i_2-l} |Z_{i_2+1}| \dots |Z_d|$$

□

The estimates above can be converted into 'logscale' estimates:

Lemma 4. *Let i_1, i_2 be successive elements of I, and let $i_1 < l < i_2$. Then the following three equations are true:*

1.

$$\frac{r(i_2)-r(i_1)}{i_2-i_1} = \log|\zeta_{i_2}| + 2^{-N+1}\log(1+c)$$

with $|c| \le \left(\max\left(\binom{d}{i_1}, \binom{d}{i_2}\right) - 1\right) R^{-1} < 2^d R^{-1}$.

2.

$$\frac{r(i_2)-r(l)}{i_2-l} \le \log|\zeta_{i_2}| + 2^{-N}\log\left(\binom{i_2-i_1}{i_2-l} + c'\right) + 2^{-N}\log|1+c|$$

where $|c| \le \left(\binom{d}{i_2} - 1\right) R^{-1} \le 2^d R^{-1}$ *and* $c' \le \left(\binom{d}{i} - \binom{i_2-i_1}{i_2-l}\right) R^{-1} < 2^d R^{-1}$.

3.

$$\frac{r(l)-r(i_1)}{l-i_1} \ge \log|\zeta_{i_2}| - 2^{-N}\log\left(\binom{i_2-i_1}{i_2-l} + c'\right) + 2^{-N}\log|1+c|$$

where $|c| \le \left(\binom{d}{i_1} - 1\right) R^{-1} \le 2^d R^{-1}$ *and* $c' \le \left(\binom{d}{i} - \binom{i_2-i_1}{l-i_1}\right) R^{-1} < 2^d R^{-1}$.

Proof of Lemma 4. By using Lemma 2 with $i = i_2$, we obtain:

$$-2^{-N}\log|\sigma_{d-i_2}(Z)| = -2^{-N}\left(\log|Z_{i_2+1}| + \dots \log|Z_d|\right) - 2^{-N}\log(|1+c''|) \tag{4}$$

Using the same lemma with $i = i_1$, we get:

$$-2^{-N}\log|\sigma_{d-i_1}(Z)| = -2^{-N}\left(\log|Z_{i_1+1}| + \dots \log|Z_d|\right) - 2^{-N}\log(|1+c''|) \tag{5}$$

Subtracting the two previous expressions and dividing by $i_2 - i_1$ we get:

$$\begin{aligned} \frac{r(i_2)-r(i_1)}{i_2-i_1} &= 2^{-N}\log|Z_{i_2}| + 2^{-N+1}\log(1+c'') \\ &= \log|\zeta_{i_2}| + 2^{-N+1}\log(1+c'') \end{aligned}$$

This shows the first part of the Lemma.

By using Lemma 3, we can also bound:

$$\begin{array}{rcl} -2^{-N} \log |\sigma_{d-l}(Z)| & \geq & -2^{-N}(i_2 - l) \log |Z_{i_2}| \\ & & -2^{-N} (\log |Z_{i_2+1}| + \ldots \log |Z_d|) \\ & & -2^{-N} \log(|\binom{i_2-i_1}{i_2-l} + c'|) \end{array} \tag{6}$$

where c' is as in Lemma 3.

We can now estimate equation (4) minus equation (6), altogether divided by $i_2 - l$:

$$\frac{r(i_2) - r(l)}{i_2 - l} \leq \log |\zeta_{i_2}| + 2^{-N} \log \left(\binom{i_2 - i_1}{i_2 - l} + c' \right) + 2^{-N} \log |1 + c|$$

We can also estimate equation (6) minus equation (5), altogether divided by $l - i_1$:

$$\frac{r(l) - r(i_1)}{l - i_1} \geq \log |\zeta_{i_2}| - 2^{-N} \log \left(\binom{i_2 - i_1}{i_2 - l} + c' \right) + 2^{-N} \log |1 + c|$$

□

3.3 A Decision Criterium

Lemma 4 can be used do decide if a point in the convex hull of g is converging to a sharp corner of the limiting convex hull or not.

Lemma 5. *Assume that*

a. $m \geq \max(i_2 - i_1)$ *when* i_1 *and* i_2 *are successive elements of* I*.*

b. $2^{-N+1} \log(2^m + 2^d R^{-1}) - 2^{-N+2} \log(1 - 2^d R^{-1}) < E < \frac{\log \rho}{2}$

c. $i < j < k$

Then,

1. *If* i *and* j *are successive elements of* I *and there is no other element of* I *between* j *and* k*, then* $\frac{r(j)-r(i)}{j-i} < \frac{r(k)-r(j)}{k-j} - E$
2. *If* i *and* k *are successive elements of* I *then* $\frac{r(j)-r(i)}{j-i} > \frac{r(k)-r(j)}{k-j} - E$

Proof of Lemma 5. Part 1: Assume that i, j are successive elements of I. Then, part 1 of Lemma 4 implies:

$$\frac{r(j) - r(i)}{j - i} \leq \log |\zeta_j| - 2^{-N+1} \log(1 - 2^d R^{-1})$$

For the evaluation of $\frac{r(k)-r(j)}{k-j}$, we have to distinguish two cases: If $k \in I$, then

$$\frac{r(k)-r(j)}{k-j} \geq \log|\zeta_k| + 2^{-N+1}\log(1-2^d R^{-1})$$

If $k \notin I$, let m be such that j and m are successive elements of I. Recall that $j < k < m$ by hypothesis. Using part 3 of Lemma 4, we get:

$$\frac{r(k)-r(j)}{k-j} \geq \log|\zeta_m| - 2^{-N}\log(2^m + 2^d R^{-1}) + 2^{-N}\log(1-2^d R^{-1})$$

In any case,

$$\begin{aligned} \frac{r(j)-r(i)}{j-i} \quad &\leq \quad \frac{r(k)-r(j)}{k-j} + \log|\zeta_j| - \log|\zeta_k| \\ &\quad +2^{-N}\log(2^m + 2^d R^{-1}) - 2^{-N+2}\log(1-2^d R^{-1}) \end{aligned}$$

We use the hypothesis $E < \frac{\log \rho}{2}$ to deduce that $\log|\zeta_j| - \log|\zeta_k| + E < -E$, and:

$$\frac{r(j)-r(i)}{j-i} < \frac{r(k)-r(j)}{k-j} - E$$

Part 2: Using Lemma 4, we have:

$$\frac{r(j)-r(i)}{j-i} \geq \log|\zeta_k| - 2^{-N}\log(2^m + 2^d R^{-1}) + 2^{-N}\log(1-2^d R^{-1})$$

$$\frac{r(k)-r(j)}{k-j} \leq \log|\zeta_k| + 2^{-N}\log(2^m + 2^d R^{-1}) - 2^{-N}\log(1-2^d R^{-1})$$

Subtracting, we obtain:

$$\begin{aligned} \frac{r(j)-r(i)}{j-i} \quad &\geq \quad \frac{r(k)-r(j)}{k-j} - 2^{-N+1}\log(2^m + 2^d R^{-1}) \\ &\quad +2^{-N+1}\log(1-2^d R^{-1}) \\ &> \quad \frac{r(k)-r(j)}{k-j} - E \end{aligned}$$

□

3.4 Proof of Proposition 2

Proof of Proposition 2. Let $N > 3+\log_2 \frac{d \log 2}{\log \rho}$. It is easy to check that $R > 2^{8d}$, hence $2^d R^{-1} < 2^{-7d}$. So we can bound:

$$2^{-N+1}\log(2^m + 2^d R^{-1}) - 2^{-N+2}\log(1-2^d R^{-1}) <$$

$$< \frac{2(m+1)\log\rho\log 2}{8d\log 2} + \frac{4\log\rho}{8d\log 2}\frac{1}{2^8-1} < \frac{\log\rho}{2}$$

Therefore, we are in the conditions 1 and 2 of Lemma 5, with $m = d$. Correctness of the algorithm 2 can be proved now by induction.

Induction Hypothesis . At step i, the list Λ contains $\Lambda_0, \ldots, \Lambda_s, \Lambda_{s+1}, \Lambda_j$ where $\Lambda_0, \ldots \Lambda_s$ are all successive elements of I and $\Lambda_{s+1}, \ldots \Lambda_j$ are not in I. (Possibly, we can have $s = j$).

The induction hypothesis is true at step 1, with $j = 0$, and $0 \in I$. At each step, there are two possibilities:

Case 1: $i \notin I$. In that case a few of the $\Lambda_{s+1}, \ldots \Lambda_j$ may be discarded; but part 1 of Lemma 5 prevents the algorithm from discarding elements of I.

Case 2: $i \in I$. In that case, part 2 of Lemma 5 guarantees that all the $\Lambda_{s+1}, \ldots \Lambda_j$ will be discarded.

Hence, the induction hypothesis is true at step $i + 1$. At step d, the last point d is added to Λ. Since $d \in I$, $\Lambda = I$.

A note on the running time: although the usual complexity of a convex hull algorithm is $O(d \log d)$ for d points in the plane, the complexity is smaller when those points are 'ordered' like ours: $(i, r(i))$. (Compare with Theorem 4.12 in [28]). Algorithm 2 has a running time of $O(d)$ operations (including a fixed number of transcendental operations). Indeed, each point is added to the list Λ precisely one time. It can be discarded only once, so the interior 'while' loop is executed at most $d - 1$ times in one execution of the algorithm.

□

4 Tangent Graeffe Iteration

4.1 Perturbation Methods, Infinitesimals, 1-Jets of Polynomials

Graeffe iteration provides the absolute values of each root in the case such roots are all of different moduli. Recovering the actual value of each root, and recovering pairs of conjugate roots or multiple roots require further work.

Many algorithms have been proposed to recover the actual roots, such as reverse Graeffe iteration, splitting algorithms. See [26] and references therein.

A possibility of theoretical interest would be to consider a perturbation of f; assume first that f is a polynomial with roots $\zeta_1, \ldots, \zeta_d$ such that $|\zeta_1| < |\zeta_2| < \cdots < |\zeta_d|$. Then, consider also the iterates of

$$f(x + \epsilon)$$

Graeffe iteration of $f(x)$ will provide $|\zeta_1|, \ldots, |\zeta_d|$, while Graeffe iteration of $f(x+\epsilon)$ will provide $|\zeta_1 - \epsilon|, \ldots, |\zeta_d - \epsilon|$. Therefore, we will be able to compute:

$$|\zeta_i|^2 - |\zeta_i - \epsilon|^2 = -2\epsilon \mathrm{Re}\ (\zeta_i) + \epsilon^2$$

thus recovering ζ_i.

As mentioned before, this is a possibility of theoretical interest only. The perturbation method above would lose half of the working precision in any reasonable implementation. Therefore, we will prefer to compute the derivative of $|\zeta - \epsilon|^2$ with respect to ϵ.

The value ϵ will be treated as an infinitesimal; therefore, instead of storing in memory a certain value $x+\epsilon\dot{x}$, we will store x and $\dot{x}$ separately. When computing some differentiable function $g(x + \epsilon\dot{x})$, we will obtain a result $g(x) + \epsilon g'(x)\dot{x}$. So we will compute $g(x)$ and $g'(x)\dot{x}$, but we will never need to assign an actual value to ϵ.

A quantity of the form $x + \epsilon\dot{x}$ is called a 1-jet. It can also be interpreted as an element of the tangent bundle of the manifold where x is supposed to live.

In this paper, we are mainly concerned with 1-jets of polynomials. We will represent degree d polynomials as points in $\mathbb{R}^{d+1}$ or $\mathbb{C}^{d+1}$. Therefore, a 1-Jet of polynomials can be represented as a point of $\mathbb{R}^{2d+2}$ or $\mathbb{C}^{2d+2}$, since we are working with a linear space.

We need the following construction from Calculus on Manifolds [1, 16]: Let G be a differentiable function from manifold X into manifold Y. Its tangent map can be written, in our 1-Jet notation, as:

$$\begin{array}{rrcl} TG: & TX & \rightarrow & TY \\ & f + \epsilon\dot{f} & \mapsto & G(f) + \epsilon DG_f \dot{f} \ , \end{array}$$

where as usual TX and TY denote the tangent bundle of X and Y respectively.

The iteration of the 1-jet of polynomials $f + \epsilon f'$ can be used to recover the actual value of the roots of f. For instance:

Example 5. Let f be a real circle free polynomial, not vanishing at 0. Consider the 1-jet $f(z+\epsilon) = f(z)+\epsilon\frac{\partial}{\partial z}f(z)$; its solutions are $\zeta_j - \epsilon$, where ζ_j are the roots of f. Let $g + \epsilon\dot{g} = TG^N(f + \epsilon\dot{f})$. Lemma 7 below will imply, in the particular case ζ_j is a real isolated root, that:

$$\zeta_j = \lim_{N\rightarrow\infty} 2^{-N} \left(\frac{|g_j|}{|g_{j-1}|}\right)^{\frac{1}{2^{N-1}}} \left(\frac{\dot{g}_j}{g_j} - \frac{\dot{g}_{j-1}}{g_{j-1}}\right)$$

In case ζ_j and $\zeta_{j+1} = \bar{\zeta}_j$ are an isolated pair of conjugate roots, the limit will be:

$$\mathrm{Re}\ \zeta_j = \lim_{N\rightarrow\infty} 2^{-N-1} \left(\frac{|g_{j+1}|}{|g_{j-1}|}\right)^{\frac{1}{2^N}} \left(\frac{\dot{g}_{j+1}}{g_{j+1}} - \frac{\dot{g}_{j-1}}{g_{j-1}}\right)$$

In the following section, we compute the tangent map of the Graeffe operator in usual and renormalized coordinates.

4.2 The Iteration

Let $f + \epsilon\dot{f}$ be a 1-Jet of polynomials. Then its Tangent Graeffe Iterate is:

$$TG(f(x) + \epsilon\dot{f}(x)) = (-1)^d \left(f(\sqrt{x}) + \epsilon\dot{f}(\sqrt{x})\right) \left(f(-\sqrt{x}) + \epsilon\dot{f}(-\sqrt{x})\right)$$

This can be rewritten as:

$$TG(f(x) + \epsilon\dot{f}(x)) = G(f) + (-1)^d\epsilon \left(f(\sqrt{x})\dot{f}(-\sqrt{x}) + f(-\sqrt{x})\dot{f}(\sqrt{x})\right)$$

Precise formulae for computing $g + \epsilon\dot{g} = TG(f + \epsilon\dot{f})$ are:

$$\begin{cases} g_i &= (-1)^{d+i}f_i^2 + 2\sum_{j\geq 1}(-1)^{d+i+j}f_{i+j}f_{i-j} \\ \\ \dot{g}_i &= 2\sum_j (-1)^{d+i+j}f_{i-j}\dot{f}_{i+j} \end{cases}$$

For an efficient root-finding algorithm the equations above need to be renormalized. At each step, this is done by replacing products and sums by their renormalized counterparts. An adjustment is necessary to pass from one renormalization level to another (division of the coordinates r by 2). Those adjustments are summarized in Algorithm 3

Algorithm 3 TangentGraeffe (N, d, r, α, $\hat{r}$, $\hat{\alpha}$)

{ N (Renormalization level) and d (degree) are integers; r and $\hat{r}$ should be real arrays, and α and $\hat{\alpha}$ should be array of modulus one complex numbers. This routine computes, in renormalized coordinates, the Tangent Graeffe Iterate of the 1-jet: $\sum_i \left(\alpha_i e^{-2^N r_i} + \epsilon\hat{\alpha}_i e^{-2^N \hat{r}_i}\right) x^i$. The coordinates of the result are given in renormalization level $N + 1$.}

$p \leftarrow 2^{N+1}$
for $i \leftarrow 0$ to d **do**
  $(s_i, \beta_i) \leftarrow (r_i, (-1)^i\alpha_i^2)$
  $(\hat{s}_i, \hat{\beta}_i) \leftarrow ((r_i + \hat{r}_i)/2 - \log(2)/p, (-1)^i\alpha_i\hat{\alpha}_i)$
  for $j \leftarrow 1$ to $\min(d - i, i)$ **do**
    $(s_i, \beta_i) \leftarrow \mathrm{RenSum}(s_i, \beta_i, (r_{i+j} + r_{i-j})/2 + \log(2)/p, (-1)^{i+j}\alpha_{i+j}\alpha_{i-j}, p)$
    $(\hat{s}_i, \hat{\beta}_i) \leftarrow \mathrm{RenSum}(\hat{s}_i, \hat{\beta}_i, (r_{i+j} + \hat{r}_{i-j})/2 + \log(2)/p, (-1)^{i+j}\alpha_{i+j}\hat{\alpha}_{i-j}, p)$
    $(\hat{s}_i, \hat{\beta}_i) \leftarrow \mathrm{RenSum}(\hat{s}_i, \hat{\beta}_i, (r_{i-j} + \hat{r}_{i+j})/2 + \log(2)/p, (-1)^{i-j}\alpha_{i-j}\hat{\alpha}_{i+j}, p)$
  end for
end for
return $(s, \beta, \hat{s}, \hat{\beta})$

4.3 Convergence Results

It is now time to show convergence of the (Renormalized) Tangent Graeffe Operator. Assume one is given a circle free polynomial f. Its roots will be ordered as $|\zeta_1| \leq |\zeta_2| \leq \cdots |\zeta_d|$. One can use the (Renormalized) Newton diagram to collect together the roots with same moduli. Those will represent single roots, multiple roots, or (in the case of real polynomials) pairs of conjugate roots or pairs of multiple conjugate roots.

Lemma 6 (Complex case). *Let f be a complex circle-free polynomial with roots $\zeta_1 \neq 0, \ldots, \zeta_d$ ordered as in Theorem 1. Let*

$$\rho \stackrel{def}{=} \min_{|\zeta_{i+1}|>|\zeta_i|} \frac{|\zeta_{i+1}|}{|\zeta_i|}$$

and let $g + \epsilon \dot{g} = (TG)^N(f + \epsilon f')$. Suppose that j and $j + d'$ are successive elements of $I = \{i : |\zeta_i| < |\zeta_{i+1}|\} \cup \{0; d\}$. Then,

$$\lim_{N\to\infty} -\frac{2^{-N}}{d'} \overline{\left(\frac{\dot{g}_{j+d'}}{g_{j+d'}} - \frac{\dot{g}_j}{g_j} \right)} = \frac{\zeta_{j+d'}}{|\zeta_{j+d'}|^2} \ .$$

Furthermore, the error is bounded by:

$$2^{d+3} \frac{d}{d'} \frac{|\zeta_d|}{|\zeta_1|} \rho^{-2^N} |\zeta_{j+d'}|^{-1}$$

Lemma 6 will be proved in Subsection 4.5.

Also, real polynomials have usually pairs of conjugate roots; they may have pairs with multiplicity. In that case, we can show that:

Lemma 7 (Real case). *Let f be a real circle-free polynomial with roots $\zeta_1 \neq 0, \ldots, \zeta_d$ ordered as in Theorem 1. Let*

$$\rho \stackrel{def}{=} \min_{|\zeta_{i+1}|>|\zeta_i|} \frac{|\zeta_{i+1}|}{|\zeta_i|}$$

and let $g + \epsilon \dot{g} = (TG)^N f + \epsilon f'$. Suppose that j and $j + d'$ are successive elements of $I = \{i : |\zeta_i| < |\zeta_{i+1}|\} \cup \{0; d\}$. Then,

$$\lim_{N\to\infty} -\frac{2^{-N}}{d'} \left(\frac{\dot{g}_{j+d'}}{g_{j+d'}} - \frac{\dot{g}_j}{g_j} \right) = \frac{Re\ \zeta_{j+d'}}{|\zeta_{j+d'}|^2} \ .$$

Furthermore, the error is bounded by:

$$2^{d+3} \frac{d}{d'} \frac{|\zeta_d|}{|\zeta_1|} \rho^{-2^N} |\zeta_{j+d'}|^{-1}$$

The proof of Lemma 7 is also postponed to subsection 4.5. Lemmas 6 and 7 can be used to recover the roots of a polynomial from the Tangent Graeffe iterates of its 1-jet:

Algorithm 4 RealRecover ($N, d, I, r, \alpha, \hat{r}, \hat{\alpha}$)

{This procedure attempts to recover the roots of the degree d real polynomial $\sum_i \alpha_i e^{-2^N r_i} x^i$. The list of sharp corners of its Newton Diagram is supposed given in $I = (I_0, \cdots, I_{1+\text{size}(I)})$. See Lemma 7 for a justification}
for $k \leftarrow 0$ to Size(I) **do**
  $d' \leftarrow I_{k+1} - I_k$
  $(b, \beta) \leftarrow \text{RenSum}\left(\hat{r}_{I_{k+1}} - r_{I_{k+1}}, \frac{\hat{\alpha}_{I_{k+1}}}{\alpha_{I_{k+1}}}, \hat{r}_{I_k} - r_{I_k}, -\frac{\hat{\alpha}_{I_k}}{\alpha_{I_k}}, 2^N\right)$
  $m \leftarrow \exp\left(2\frac{r_{I_{k+1}} - r_{I_k}}{d'}\right)$
  $x \leftarrow -\beta \frac{2^{-N}}{d'} m \exp -2^N b$
  if $I_{k+1} - I_k$ is even and $m > |x|^2$ **then**
    $y \leftarrow \sqrt{m - |x|^2}$
  else
    $x \leftarrow m \frac{x}{|x|}$
    $y \leftarrow 0$
  end if
  for $j \leftarrow 0$ to $I_{k+1} - I_k - 1$ **do**
    $\zeta_{I_k + j + 1} = x + (-1)^j y$
  end for
end for
return ζ

Algorithm 5 ComplexRecover ($N, d, I, r, \alpha, \hat{r}, \hat{\alpha}$)

{This procedure attempts to recover the roots of the degree d complex polynomial $\sum_i \alpha_i e^{-2^N r_i} x^i$. The list of sharp corners of its Newton Diagram is supposed given in I. See Lemma 6 for a justification}
for $k \leftarrow 0$ to Size(I) **do**
  $d' \leftarrow I_{k+1} - I_k$
  $(b, \beta) \leftarrow \text{RenSum}\left(\hat{r}_{I_{k+1}} - r_{I_{k+1}}, \frac{\hat{\alpha}_{I_{k+1}}}{\alpha_{I_{k+1}}}, \hat{r}_{I_k} - r_{I_k}, -\frac{\hat{\alpha}_{I_k}}{\alpha_{I_k}} 2^N, \right)$
  $m \leftarrow \exp\left(2\frac{r_{I_{k+1}} - r_{I_k}}{d'}\right)$
  $x \leftarrow -\bar{\beta} \frac{2^{-N}}{d'} m \exp -2^N b$
  for $j \leftarrow 0$ to $I_{k+1} - I_k - 1$ **do**
    $\zeta_{I_k + j + 1} = x$
  end for
end for
return ζ

4.4 The Main Algorithm

We can now state the algorithm of Theorem 1. We start with a fixed, arbitrary value for $\rho(f) = \max_{|\zeta_i| > |\zeta_j|} \frac{\zeta_i}{\zeta_j}$. Proposition 2 guarantees that if

$$N > 3 + \log_2 \frac{d \log 2}{\log \rho(f)} ,$$

then after the N-th iterate the convex hull of the Newton Diagram of f is computed correctly.

Algorithm 6 Solve (d, f, isreal)

{It is assumed here that f is a degree d, circle-free real or complex polynomial. In the general case, one should first find and output the trivial (0 and ∞) roots of f, then deflate f. After that, one should perform a random real (resp. complex) conformal transform on f so it becomes circle-free}

for $i \leftarrow 0$ to d **do**
  if $f_i \neq 0$ **then**
    $\alpha_i \leftarrow f_i/|f_i|$
  else
    $\alpha_i \leftarrow 1$
  end if
  $r_i \leftarrow -\log|f_i|$
end for
for $i \leftarrow 0$ to $d-1$ **do**
  $f'_i \leftarrow (i+1)f_{i+1}$
  if $f'_i \neq 0$ **then**
    $\hat{\alpha}_i \leftarrow f'_i/|f'_i|$
  else
    $\hat{\alpha}_i \leftarrow 1$
  end if
  $\hat{r}_i \leftarrow -\log|f'_i|$
end for
$N \leftarrow 0$
$\rho = 2$
loop
  $r, \alpha, \hat{r}, \hat{\alpha} \leftarrow$ TangentGraeffe $(N, d, r, \alpha, \hat{r}, \hat{\alpha})$
  $N \leftarrow N+1$
  $I \leftarrow$ Convex Hull $(N,\ d,\ r,\ \rho)$
  if isreal **then**
    $\zeta \leftarrow$ RealRecover $(\ N, d, I, r, \alpha, \hat{r}, \hat{\alpha}\ )$
  else
    $\zeta \leftarrow$ ComplexRecover $(\ N, d, I, r, \alpha, \hat{r}, \hat{\alpha}\ )$
  end if
  Output $\zeta_1, \cdots \zeta_d$
  if $N > 3 + \log_2 \frac{d \log 2}{\log \rho}$ **then**
    { Proposition 2 implies that at this point, I is indeed correct for all the polynomials with separation ration $\geq \rho$. Therefore, it is time to decrease ρ.}
    $\rho \leftarrow \sqrt{\rho}$
  end if
end loop

After the N-th iteration, convergence is guaranteed by the following bounds: According to Lemma 4, at the execution of algorithm Complex Recover (resp. Real Recover),

$$\left| \exp\left(\frac{2}{I_{k+1} - I_k} (f_{I_{k+1}} - f_{I_k}) \right) \right| = |\zeta_{I_{k+1}}|(1 + \delta_1)$$

where $|\delta_1| \le e^{2^{-N+1} \log 1 + 2^d \rho^{-2^N}}$.

Introducing the error bound of Lemma 6, one gets in the complex case:

$$-\frac{2^{-N}}{d'} \overline{(a - b)} = \frac{\zeta_{I_{k+1}}}{|\zeta_{I_{k+1}}|^2}(1 + \delta_2)$$

with $|\delta_2| < 2^{d+3} \frac{d}{d'} \frac{|\zeta_d|}{|\zeta_1|} \rho^{-2^N}$. Therefore,

$$|\zeta_{I_k + j + 1} - \bar{x}| \le \delta |\zeta_{I_k + j + 1}|$$

where $\delta < (1 + \delta_1)^2(1 + \delta_2) - 1$.

The real case is analogous. According to Lemma 7,

$$-\frac{2^{-N}}{d'}(a - b) = \frac{\text{Re } \zeta_{I_{k+1}}}{|\zeta_{I_{k+1}}|^2}(1 + \delta_2)$$

so that

$$|\text{Re } \zeta_{I_k + j + 1} - x| < \delta |\zeta_{I_k + j + 1}|$$

and

$$|\text{Im } \zeta_{I_k + j + 1} - \sqrt{1 - x^2}| < \delta' |\zeta_{I_k + j + 1}|$$

where $\delta' = \delta + O(\delta^2)$

In both cases, $|\delta|$ and eventually $|\delta'|$ are dominated by ρ^{-2^N}. Since

$$A \rho^{-2^N} = 2^{\log_2 A - 2^N \log_2 \rho} 2^{-2^{N-C}} ,$$

the bound in Theorem 1 follows.

4.5 Proof of Lemmas 6 and 7

Consider the 1-jet of degree d polynomials $f + \epsilon \dot{f}$, with solutions $\zeta_i + \epsilon \dot{\zeta}_i$. After N steps of Tangent Graeffe Iteration, we obtain a 1-jet of polynomials

$$g + \epsilon \dot{g} = (TG)^N (f + \epsilon \dot{f}) .$$

Since the differential of the transformation $P^N : \zeta \longmapsto z = \zeta^{2^N}$ gives

$$DP^N : \zeta + \epsilon \dot{\zeta} \longmapsto (\zeta_j)^{2^N} + \epsilon 2^N (\zeta_j)^{2^N} \frac{\dot{\zeta}_j}{\zeta_j} ,$$

it is clear that the roots $z_j + \epsilon \dot{z}_j$ of $g + \epsilon \dot{g}$ will be

$$(\zeta_j)^{2^N} + \epsilon 2^N (\zeta_j)^{2^N} \frac{\dot{\zeta}_j}{\zeta_j} \ .$$

We can now compute the derivative at $\epsilon = 0$ of $g_i + \epsilon g_i = \sigma_{d-i}(z + \epsilon \dot{z})$.

Let's denote by $z_{\hat{j}}$ the vector $(z_1, \cdots, \widehat{z_j}, \cdots, z_d) \in \mathbb{C}^{d-1}$.

Take $i \in I$ and notice that

$$\begin{aligned} \left.\frac{\partial}{\partial \epsilon}\right|_{\epsilon=0} \sigma_{d-i}(z + \epsilon \dot{z}) &= \sum_j \dot{z}_j \sigma_{d-i-1}(z_{\hat{j}}) \\ &= \sum_j \frac{\dot{z}_j}{z_j} z_j \sigma_{d-i-1}(z_{\hat{j}}) \end{aligned}$$

Thus,

$$\begin{aligned} \left.\frac{\partial}{\partial \epsilon}\right|_{\epsilon=0} \frac{\sigma_{d-i}(z + \epsilon \dot{z})}{\sigma_{d-i}(z)} &= \sum_j \frac{\dot{z}_j}{z_j} \frac{z_j \sigma_{d-i-1}(z_{\hat{j}})}{\sigma_{d-i}(z)} \\ &= \left(\sum_{j>i} + \sum_{j \le i} \right) \frac{\dot{z}_j}{z_j} \frac{z_j \sigma_{d-i-1}(z_{\hat{j}})}{\sigma_{d-i}(z)} \ . \end{aligned}$$

Due to Lemma 2

$$\sum_{j>i} \frac{\dot{z}_j}{z_j} \frac{z_j \sigma_{d-i-1}(z_{\hat{j}})}{\sigma_{d-i}(z)} = \sum_{j>i} \frac{\dot{z}_j}{z_j} (1 + \eta_j)$$

and

$$\sum_{j \le i} \frac{\dot{z}_j}{z_j} \frac{z_j \sigma_{d-i-1}(z_{\hat{j}})}{\sigma_{d-i}(z)} = \sum_{j \le i} \frac{\dot{z}_j}{z_j} \frac{z_j}{z_{i+1}} (1 + \eta_j) \ ,$$

where

$$|\eta_j| \le 4 \binom{d}{i} R^{-1} \ .$$

Although this estimate is by no means sharp, it suffices for our purposes.

Using that $i \in I$ and hence $|z_j / z_{i+1}| < R^{-1}$ for $i \le j$, we get

$$\begin{aligned} \left| \left.\frac{\partial}{\partial \epsilon}\right|_{\epsilon=0} \frac{\sigma_{d-i}(z + \epsilon \dot{z})}{\sigma_{d-i}(z)} - \sum_{j>i} \frac{\dot{z}_j}{z_j} \right| &\le \max \left| \frac{\dot{z}_j}{z_j} \right| \frac{4\binom{d}{i}(d-i) + i\left(1 + 4\binom{d}{i} R^{-1}\right)}{R} \\ &< \max \left| \frac{\dot{z}_j}{z_j} \right| d 2^{d+2} R^{-1} \end{aligned}$$

Therefore, if we take the logarithmic derivative of the expression

$$(g+\epsilon\dot{g}) = (TG)^N(f+\epsilon\dot{f})$$

and evaluate at a successive couple of elements $i_1, i_2 \in I$ we get

$$\left|\left(\frac{\dot{g}_{i_1}}{g_{i_1}} - \frac{\dot{g}_{i_2}}{g_{i_2}}\right) + \sum_{i_1<j\le i_2} 2^N \frac{1}{\zeta_j}\right| < \frac{2^{N+d+3}d}{R} \max_l \left|\frac{1}{\zeta_l}\right| .$$

Now let's assume we are under the hypothesis of Lemma 6. Then, since i_1 and i_2 are consecutive, it follows that

$$\sum_{i_1<j\le i_2} \zeta_j^{-1} = \sum_{i_1<j\le i_2} \frac{\overline{\zeta_j}}{|\zeta_j|^2} = (i_2-i_1)\frac{\overline{\zeta_{i_2}}}{|\zeta_{i_2}|^2}$$

and so

$$\left|\frac{2^{-N}}{i_2-i_1}\left(\frac{\dot{g}_{i_2}}{g_{i_2}} - \frac{\dot{g}_{i_1}}{g_{i_1}}\right)|\zeta_{i_2}|^2 - \overline{\zeta_{i_2}}\right| |\zeta_{i_2}|^{-1} < \frac{2^{d+3}d}{R(i_2-i_1)} \max_{r,s} \frac{|\zeta_r|}{|\zeta_s|} . \tag{7}$$

On the other hand, if we are under the hypothesis of Lemma 7, we get

$$\sum_{i_1<j\le i_2} \zeta_j^{-1} = (i_2-i_1)\frac{\mathrm{Re}\ \zeta_{i_2}}{|\zeta_{i_2}|^2} .$$

Thus,

$$\left|\frac{2^{-N}}{i_2-i_1}\left(\frac{\dot{g}_{i_2}}{g_{i_2}} - \frac{\dot{g}_{i_1}}{g_{i_1}}\right)|\zeta_{i_2}|^2 - \mathrm{Re}\ \zeta_{i_2}\right| |\zeta_{i_2}|^{-1} < \frac{2^{d+3}d}{R(i_2-i_1)} \max_{r,s} \frac{|\zeta_r|}{|\zeta_s|} . \tag{8}$$

In either case, we have that each right hand side of equations (7) and (8) is bounded by

$$2^{d+3}\frac{d}{R(i_2-i_1)}\frac{|\zeta_d|}{|\zeta_1|} .$$

Now, using the fact that the separation radius at the N-th step

$$R = \rho^{2^N} ,$$

where ρ is the separation radius of the original roots of f before applying Graeffe, we get that

$$\frac{2^{d+3}d}{R(i_2-i_1)}\frac{|\zeta_d|}{|\zeta_1|} = 2^{d+3}\frac{d}{i_2-i_1}\frac{|\zeta_d|}{|\zeta_1|}\rho^{-2^N} \longrightarrow 0 \text{ , as } N \longrightarrow \infty .$$

5 Numerical Results and Final Remarks

5.1 Numerical Results

A polynomial solver based on the algorithms above was implemented and tested under IEEE 754 double and double-extended arithmetic. The results below are intended to make a case in favor of the stability and the practical feasibility of Renormalized Tangent Graeffe Iteration.

The first set of tests was designed to measure the performance of our algorithm for large degree polynomials. The test polynomials are pseudo-random real (Table 1 and Figure 3) and complex (Table 2 and Figure 4) polynomials, under the $U(2)$-invariant probability measure [15, 32, 20]. Under this probability measure, random polynomials are well-conditioned in average.

The results were certified using alpha-theory [36, 31, 18]. The running-time (certification excluded) was compared to the code of Jenkins and Traub [12, 11] for the values where this code succeeds.

Running time is measured in user-time seconds of a Pentium-133 computer running Linux and the gcc compiler.

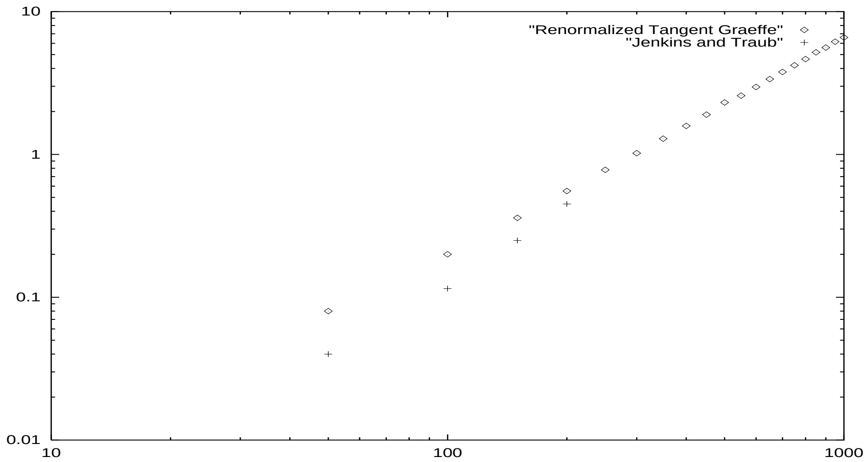

Figure 3: Real pseudo-random polynomials

In the second set of experiments, we tried to check the behavior of Renormalized Tangent Graeffe Iteration in the presence of very badly conditioned polynomials. The test polynomials are Wilkinson' s 'perfidious' polynomials [38]

$$p_d(x) = (x-1)(x-2)\cdots(x-d)$$

and Chebyshev polynomials (Table 3).

$$T_d = \prod_{0 \le m < d} \left(x - \cos(\frac{\pi}{2d} + \frac{\pi}{d} m) \right)$$

Running time (s)											
Degree	Algorithm	Seed									
		0	1	2	3	4	5	6	7	8	9
50	Graeffe	0.08	0.08	0.08	0.07	0.09	0.08	0.08	0.08	0.08	0.07
	J-T	0.03	0.02	0.04	0.02	0.05	0.05	0.07	0.04	0.04	0.03
100	Graeffe	0.20	0.20	0.20	0.21	0.21	0.21	0.19	0.20	0.19	0.20
	J-T	0.12	0.11	0.12	0.09	0.12	0.12	0.14	0.09	0.10	0.10
150	Graeffe	0.36	0.36	0.36	0.36	0.37	0.35	0.36	0.37	0.35	0.36
	J-T	0.26	0.26	0.28	0.24	0.23	0.24	0.27	0.20	0.26	0.24
200	Graeffe	0.56	0.54	0.56	0.56	0.55	0.55	0.56	0.55	0.56	0.55
	J-T	0.53	0.42	0.51	0.35	0.43	0.38	0.54	0.40	0.47	0.36
250	Graeffe	0.77	0.78	0.77	0.78	0.78	0.78	0.78	0.78	0.77	0.76
300	Graeffe	1.02	1.02	1.01	1.02	1.02	1.00	1.02	1.01	1.01	1.03
350	Graeffe	1.29	1.29	1.29	1.29	1.30	1.29	1.29	1.29	1.28	1.29
400	Graeffe	1.59	1.59	1.57	1.58	1.58	1.60	1.57	1.59	1.59	1.58
450	Graeffe	1.90	1.90	1.89	1.91	1.89	1.89	1.90	1.90	1.91	1.91
500	Graeffe	2.32	2.27	2.31	2.30	2.31	2.31	2.30	2.29	2.32	2.34
550	Graeffe	2.57	2.58	2.58	2.58	2.59	2.59	2.58	2.59	2.57	2.59
600	Graeffe	2.96	2.98	2.97	2.96	2.95	2.97	2.96	2.97	2.98	2.96
650	Graeffe	3.38	3.39	3.37	3.36	3.37	3.34	3.38	3.37	3.37	3.37
700	Graeffe	3.78	3.78	3.78	3.80	3.79	3.77	3.78	3.80	3.78	3.77
750	Graeffe	4.21	4.22	4.21	4.21	4.19	4.20	4.19	4.22	4.20	4.21
800	Graeffe	4.66	4.66	4.65	4.65	4.64	4.64	4.63	4.65	4.65	4.65
850	Graeffe	5.18	5.20	5.18	5.17	5.23	5.18	5.19	5.20	5.19	5.17
900	Graeffe	5.61	5.61	5.58	5.59	5.60	5.57	5.56	5.60	5.60	5.60
950	Graeffe	6.16	6.16	6.15	6.16	6.13	6.14	6.16	6.18	6.14	6.18
1000	Graeffe	6.60	6.58	6.58	6.58	6.59	6.61	6.59	6.60	6.58	6.59

Table 1: Real pseudo-random polynomials

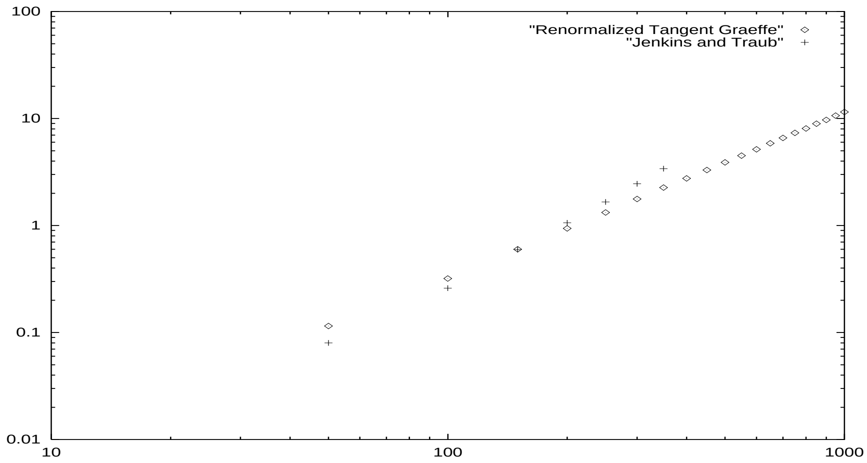

Figure 4: Complex pseudo-random polynomials

Running time (s)											
Degree	Algorithm	Seed									
		0	1	2	3	4	5	6	7	8	9
50	Graeffe	0.13	0.12	0.12	0.13	0.12	0.10	0.11	0.10	0.11	0.11
	J-T	0.07	0.08	0.09	0.09	0.07	0.08	0.07	0.08	0.07	0.09
100	Graeffe	0.34	0.32	0.32	0.32	0.33	0.33	0.30	0.33	0.31	0.32
	J-T	0.25	0.28	0.26	0.25	0.28	0.26	0.26	0.27	0.26	0.25
150	Graeffe	0.60	0.60	0.61	0.60	0.61	0.60	0.60	0.61	0.60	0.60
	J-T	0.55	0.60	0.58	0.66	0.61	0.60	0.60	0.59	0.59	0.59
200	Graeffe	0.95	0.95	0.97	0.94	0.96	0.94	0.92	0.94	0.94	0.93
	J-T	1.13	1.17	1.07	1.04	1.09	1.06	1.05	1.06	1.03	1.05
250	Graeffe	1.33	1.34	1.30	1.32	1.35	1.32	1.32	1.32	1.35	1.33
	J-T	1.63	1.68	1.64	1.68	1.66	1.70	1.66	1.63	1.63	1.66
300	Graeffe	1.77	1.76	1.77	1.78	1.77	1.76	1.77	1.75	1.79	1.76
	J-T	2.34	2.49	2.43	2.44	2.50	2.46	2.48	2.45	2.36	2.51
350	Graeffe	2.26	2.30	2.26	2.25	2.28	2.34	2.08	2.36	2.27	2.26
	J-T	3.30	3.41	3.39	3.34	3.36	3.72	3.51	3.45	3.35	3.52
400	Graeffe	2.74	2.75	2.77	2.76	2.77	2.76	2.75	2.77	2.74	2.74
450	Graeffe	3.27	3.28	3.28	3.30	3.38	3.30	3.30	3.28	3.34	3.30
500	Graeffe	3.92	3.87	3.89	3.89	3.88	3.89	3.91	3.87	3.90	3.91
550	Graeffe	4.49	4.48	4.51	4.52	4.50	4.52	4.51	4.49	4.50	4.48
600	Graeffe	5.16	5.14	5.17	5.18	5.15	5.14	5.15	5.15	5.18	5.13
650	Graeffe	5.89	5.88	5.87	5.83	5.83	5.86	5.89	5.84	5.89	5.86
700	Graeffe	6.62	6.59	6.59	6.58	6.59	6.61	6.59	6.59	6.58	6.62
750	Graeffe	7.37	7.42	7.40	7.32	7.48	7.27	7.30	7.30	7.43	7.31
800	Graeffe	8.10	8.06	8.07	8.03	8.10	8.07	8.10	8.06	8.11	8.09
850	Graeffe	8.96	8.94	8.92	8.95	8.97	8.94	8.92	8.92	8.95	8.96
900	Graeffe	9.72	9.70	9.70	9.70	9.72	9.72	9.71	9.70	9.67	9.71
950	Graeffe	10.63	10.67	10.65	10.64	10.67	10.65	10.67	10.61	10.65	10.60
1000	Graeffe	11.49	11.47	11.54	11.46	11.50	11.49	11.54	11.50	11.51	11.50

Table 2: Complex pseudo-random polynomials

Perfidious polynomials		
Degree	Algorithm	Error
10	Graeffe	5.123013×10^{-12}
	J-T	4.859935×10^{-11}
15	Graeffe	3.968295×10^{-08}
	J-T	5.508868×10^{-09}
20	Graeffe	1.780775×10^{-03}
	J-T	1.275754×10^{-04}

Chebyshev polynomials		
Degree	Algorithm	Error
10	Graeffe	8.790711×10^{-16}
	J-T	8.790711×10^{-16}
15	Graeffe	2.169163×10^{-15}
	J-T	2.169163×10^{-15}
20	Graeffe	1.903848×10^{-14}
	J-T	1.278977×10^{-13}
25	Graeffe	1.266375×10^{-11}
	J-T	1.663025×10^{-11}
30	Graeffe	5.511325×10^{-11}
	J-T	3.301608×10^{-10}
35	Graeffe	5.708941×10^{-09}
	J-T	3.840000×10^{-08}

Table 3: Perfidious and Chebyshev polynomials

The error of the solutions of the perfidious polynomials is measured as $\max |\zeta - \text{round } \zeta|$, ζ a 'solution' found by the program. Similarly, the error in Chebyshev polynomials is measured as $\max |m - \text{ round } m|$ where $m = \frac{d \arccos \zeta - \frac{\pi}{2}}{\pi}$ and ζ a 'solution' found by the program. Again, those results are compared to the ones provided by the software by Jenkins and Traub.

5.2 Further Practical Remarks

- Graeffe process (and hence our algorithm) is known to be parallelizable [10, 29].
- The algorithm presented here needs only $O(d)$ memory storage; therefore, all intermediate computations for a reasonable degree d fit into the 'cache' memory of modern computers.
- Use of higher precision may be required to handle very badly conditionned polynomials; as a matter of fact, polynomials of that sort are only meaningful if their coefficients are known with sufficiently high accuracy. For example, when they are obtained by symbolic manipulation.

Acknowledgements

JPZ's work was supported in part by CNPq, through grant MA 521329/94-9, and by PRONEX grant 76.97.1008-00. GM's work was supported by CNPq through grant MA 520423/96-8 and FAPERJ E-26/170.027/98. J.P.Z. also acknowledges the participation at the MGSS which was supported through grant NSF DMS97-09320.

References

[1] V. Arnold, *Méthodes Mathématiques de la Mécanique Classique*, Mir, Moscow, 1976.

[2] D. Bini and V. Y. Pan, *Graeffe's, Chebyshev-like, and Cardinal's processes for splitting a polynomial into factors*, J. Complexity, 12 (1996), pp. 492–511. Special issue for the Foundations of Computational Mathematics Conference (Rio de Janeiro, 1997).

[3] J.-P. Dedieu, *À propos de la méthode de Dandelin-Graeffe*, C. R. Acad. Sci. Paris Sér. I Math., 309 (1989), pp. 1019–1022.

[4] J. P. Dedieu, X. Gourdon, and J. C. Yakoubsohn, *Computing the distance from a point to an algebraic hypersurface*, in The mathematics of numerical analysis (Park City, UT, 1995), vol. 32 of Lectures in Appl. Math., Amer. Math. Soc., Providence, RI, 1996, pp. 285–293.

[5] J. W. Demmel, *Applied numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[6] A. A. Grau, *On the reduction of number range in the use of the Graeffe process*, J. Assoc. Comput. Mach., 10 (1963), pp. 538–544.

[7] P. Henrici, *Applied and computational complex analysis. Vol. 3*, Pure and Applied Mathematics, John Wiley & Sons Inc., New York, 1986. Discrete Fourier analysis—Cauchy integrals—construction of conformal maps—univalent functions, A Wiley-Interscience Publication.

[8] N. J. Higham, *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

[9] A. S. Householder, *Dandelin, Lobačevskiĭ, or Graeffe?*, Amer. Math. Monthly, 66 (1959), pp. 464–466.

[10] P. Jana and B. Sinha, *Fast parallel algorithms for Graeffe' s root squaring*, Comput. Math. Appl., 35 (1998), pp. 71–80.

[11] M. A. Jenkins, *Algorithm 493 – Zeros of a real polynomial [C2]*, ACM Transactions on Mathematical Software, 1 (1975), pp. 178–189.

[12] M. A. Jenkins and J. F. Traub, *Algorithm 419 – Zeros of a complex polynomial [C2]*, Comm. of the ACM, 15 (1972), pp. 97–99.

[13] L. V. Kantorovich and G. P. Akilov, *Functional analysis*, Pergamon Press, Oxford, second ed., 1982. Translated from the Russian by Howard L. Silcock.

[14] P. Kirrinis, *Partial fraction decomposition in* $\mathbb{C}(z)$ *and simultaneous newton iteration for factorization in* $\mathbb{C}[z]$. Preprint, Bonn, (1995).

[15] E. Kostlan, *Random polynomials and the statistical fundamental theorem of algebra*, 1987. Preprint, MSRI, 1987.

[16] S. Lang, *Differentiable Manifolds*, Addison-Wesley Pub. Co., Reading, Mass., 1972.

[17] R. S. MacKay, *Renormalisation in area-preserving maps*, vol. 6 of Advanced Series in Nonlinear Dynamics, World Scientific Publishing Co. Inc., River Edge, NJ, 1993.

[18] G. Malajovich, *On generalized Newton algorithms: quadratic convergence, path-following and error analysis*, Theoret. Comput. Sci., 133 (1994), pp. 65–84. Selected papers of the Workshop on Continuous Algorithms and Complexity (Barcelona, 1993).

[19] G. Malajovich and J. P. Zubelli, *A fast and stable algorithm for splitting polynomials*, Comput. Math. Appl., 33 (1997), pp. 1–23.

[20] G. Malajovich and J. P. Zubelli, *On the geometry of Graeffe iteration*, 1997. Informes de Matemática Série B-118, IMPA.

[21] C. T. McMullen, *Complex dynamics and renormalization*, vol. 135 of Annals of Mathematics Studies, Princeton University Press, Princeton, NJ, 1994.

[22] C. A. Neff and J. H. Reif, *An efficient algorithm for the complex roots problem*, J. Complexity, 12 (1996), pp. 81–115.

[23] A. Ostrowski, *Recherches sur la méthode de Graeffe et les zéros des polynomes et des séries de Laurent*, Acta Math., 72 (1940), pp. 99–155.

[24] ——, *Recherches sur la méthode de Graeffe et les zéros des polynomes et des séries de Laurent. Chapitres III et IV*, Acta Math., 72 (1940), pp. 157–257.

[25] V. Y. Pan, *Optimal and nearly optimal algorithms for approximating polynomial zeros*, Comput. Math. Appl., 31 (1996), pp. 97–138.

[26] V. Y. Pan, *Solving a polynomial equation: some history and recent progress*, SIAM Rev., 39 (1997), pp. 187–220.

[27] V. Y. Pan, M.-H. Kim, A. Sadikou, X. Huang, and A. Zheng, *On isolation of real and nearly real zeros of a univariate polynomial and its splitting into factors*, J. Complexity, 12 (1996), pp. 572–594. Special issue for the Foundations of Computational Mathematics Conference (Rio de Janeiro, 1997).

[28] F. P. Preparata and M. I. Shamos, *Computational geometry*, Texts and Monographs in Computer Science, Springer-Verlag, New York, 1985. An introduction.

[29] T. A. Rice and L. H. Jamieson, *A highly parallel algorithm for root extraction*, IEEE Trans. Comput., 38 (1989), pp. 443–449.

[30] A. Schönhage, *Equation solving in terms of computational complexity*, in Proceedings of the International Congress of Mathematicians, Vol. 1, 2 (Berkeley, Calif., 1986), Providence, RI, 1987, Amer. Math. Soc., pp. 131–153.

[31] M. Shub and S. Smale, *Complexity of Bézout's theorem. I. Geometric aspects*, J. Amer. Math. Soc., 6 (1993), pp. 459–501.

[32] M. Shub and S. Smale, *Complexity of Bezout's theorem. II. Volumes and probabilities*, in Computational algebraic geometry (Nice, 1992), vol. 109 of Progr. Math., Birkhäuser Boston, Boston, MA, 1993, pp. 267–285.

[33] M. Shub and S. Smale, *Complexity of Bezout's theorem. III. Condition number and packing*, J. Complexity, 9 (1993), pp. 4–14. Festschrift for Joseph F. Traub, Part I.

[34] M. Shub and S. Smale, *Complexity of Bezout's theorem. V. Polynomial time*, Theoret. Comput. Sci., 133 (1994), pp. 141–164. Selected papers of the Workshop on Continuous Algorithms and Complexity (Barcelona, 1993).

[35] M. Shub and S. Smale, *Complexity of Bezout's theorem. IV. Probability of success; extensions*, SIAM J. Numer. Anal., 33 (1996), pp. 128–148.

[36] S. Smale, *Newton's method estimates from data at one point*, in The merging of disciplines: new directions in pure, applied, and computational mathematics (Laramie, Wyo., 1985), Springer, New York, 1986, pp. 185–196.

[37] M. M. Vainberg, *Variational methods for the study of nonlinear operators*, Holden-Day Inc., San Francisco, Calif., 1964. With a chapter on Newton's method by L. V. Kantorovich and G. P. Akilov. Translated and supplemented by Amiel Feinstein.

[38] J. H. Wilkinson, *The perfidious polynomial*, in Studies in numerical analysis, vol. 24 of MAA Stud. Math., Math. Assoc. America, Washington, DC, 1984, pp. 1–28.

Gregorio Malajovich
Departamento de Matemática Aplicada
Universidade Federal do Rio de Janeiro
Caixa Postal 68530 – CEP 21945
Rio de janeiro – RJ – BRAZIL
gregorio@labma.ufrj.br
http://www.labma.ufrj.br/~gregorio

Jorge P. Zubelli
IMPA - CNPq
Est. D. Castorina 110
Rio de Janeiro, RJ 22460-320, BRAZIL
zubelli@impa.br
http://www.impa.br/~zubelli