# Strong Solution to Smale's 17th Problem for Strongly Sparse Systems

Paula Burkhardt

Pomona College and Texas A&M University

July 24, 2014

**Abstract**

Smale's 17th problem asks whether one can deterministically approximate a single root of a system of polynomials, in polynomial-time on average. The best recent results are probabilistic polynomial-time algorithms, so Smale's 17th Problem has not yet been fully solved. We give a much faster deterministic algorithm for the special case of binomial systems, and certain systems of binomials and trinomials. Our approach is also a stepping stone to harder variants of Smale's 17th Problem, such as approximating roots near a query point or approximating a single real root.

## 1 Introduction

Smale's 17th Problem asks: *Does there exist a deterministic algorithm which approximates a root of a system on polynomials and runs in polynomial time on average?* We present polynomial-time (or faster) algorithms for the cases of univariate binomials and systems of binomials, and a partial algorithm for trinomials. Our approach to this problem is novel in that it focuses on the number of monomial terms of a given polynomial. The coefficients of our polynomials are random variables whose coefficients are selected from a complex normal distribution (we discuss the significance of these distributions later in the paper). We begin with the question of approximating the $d$th root of a complex number and reduce more elaborate systems to this case.

### 1.1 Some Definitions

Here we recall some definitions due to Smale. Let $||\cdot||$ denote the Euclidean norm or the operator norm as appropriate. Then we have:

**Definition – Approximate Root (Smale [1986]).** *Suppose $f : \mathbb{C}^n \to \mathbb{C}^n$ is a multivariate polynomial. Let $z \in \mathbb{C}^n$ be a point such that*

$$||\zeta - N_f^k(z)|| \leq \frac{1}{2^{2^k} - 1}||\zeta - z||$$

*where $N_f$ is the Newton operator, $z \mapsto z - Df(z)^{-1}f(z)$, and $\zeta$ is an actual root of $f$. Then we say $z$ is an approximate root of $f$ with associated true root $\zeta$.*

We work with polynomials whose coefficients are random variables selected from complex normal distributions. We will discuss the question of selecting appropriate probability density functions in greater detail later in this paper.

## 1.2 $\alpha$-Theory

Here we recall some results of Smale.

**Definition – $\alpha$, $\beta$, and $\gamma$ (Smale [1986]).** *For an analytic function $f : \mathbb{C}^n \to \mathbb{C}^n$ and $z \in \mathbb{C}^n$ define the following:*

$$\gamma(f, z) := \sup_{k \geq 2} \left|\left| \frac{f'(z)^{-1}f^{(k)}(z)}{k!} \right|\right|^{\frac{1}{k-1}}$$

$$\beta(f, z) := ||f'(z)^{-1}f(z)||$$

*and*

$$\alpha(f, z) := \beta(f, z)\gamma(f, z)$$

These items are useful for providing lower bounds on the possible distance of an approximate root from its associated true root, as we see below.

**$\gamma$-Theorem (Smale [1986]).** *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be analytic, and suppose $\zeta \in \mathbb{C}^n$ such that $f(\zeta) = 0$. Then if*

$$||z - \zeta|| \leq \frac{3 - \sqrt{7}}{2\gamma(f, \zeta)}$$

*$z$ is an approximate root of $f$ with associated true root $\zeta$.*

A somewhat stronger result by Smale gives us that

**$\alpha$-Theorem (Smale [1986]).** *There exists a universal constant $\alpha_0$ such that if $z \in \mathbb{C}^n$ with $\alpha(f, z) < \alpha_0$ then $z$ is an approximate root of $f$.*

Smale gives a lower bound for $\alpha_0$ as 0.1370707, though less conservative bounds have since been found.

### 1.3 Some Useful Results

We would like to come up with a lower bound on $\frac{3-\sqrt{7}}{2\gamma(f,\zeta)}$ in order to approximate our roots.

**Lemma.** *For any univariate polynomial $f(x_1) = c_1 x_1^{a_1} + \ldots + c_t x_1^{a_t}$ where $c_1, \ldots, c_t \in \mathbb{C}^*$ and $a_1, \ldots, a_t \in \mathbb{N}$ with $0 < a_1 < \ldots < a_t$ we have that $\gamma(f,z) \leq \left|\frac{a_t - 1}{2z}\right|$ for all $z \in \mathbb{C}$.*

*Proof.*

$$\gamma(f,x) = \sup_{k \geq 2} \left| \frac{c_1 a_1 \cdots (a_1 - k + 1) x^{a_1 - k} + \ldots + c_t a_a \cdots (a_t - k + 1) x^{a_t - k}}{k!(c_1 a_1 x^{a_1 - 1} + \ldots + c_t a_t x^{a_t - 1})} \right|^{\frac{1}{k-1}}$$

$$= \sup_{k \geq 2} \left| \left( \frac{1}{k! x^{k-1}} \right) \left( \frac{c_1 a_1 \cdots (a_1 - k + 1) + \ldots + c_t a_t \cdots (a_t - k + 1) x^{a_t - a_1}}{c_1 a_1 + \ldots + c_t a_t x^{a_t - a_1}} \right) \right|^{\frac{1}{k-1}}$$

$$\leq \sup_{k \geq 2} \left| \left( \frac{1}{k! x^{k-1}} \right) \left( \frac{c_1 a_1 (a_t - 1)^{k-1} + \ldots + c_t a_t (a_t - 1)^{k-1} x^{a_t - a_1}}{c_1 a_1 + \ldots + c_t a_t x^{a_t - a_1}} \right) \right|^{\frac{1}{k-1}}$$

$$\leq \sup_{k \geq 2} \left| \frac{(a_t - 1)^{k-1}}{2^{k-1} x^{k-1}} \right|^{\frac{1}{k-1}} = \left| \frac{a_t - 1}{2x} \right|$$

□

This upper bound on $\gamma$ gives us a lower bound on $\frac{3-\sqrt{7}}{2\gamma(f,c^{\frac{1}{d}})}$ where $f(x_1) := x_1^d - c$ is a univariate polynomial:

**Example.** *If $|c| > 1$ then if $z \in \mathbb{C}$ satisfies*

$$|z - c^{\frac{1}{d}}| \leq \frac{1}{3d} \leq \frac{3-\sqrt{7}}{d-1} |c^{\frac{1}{d}}|$$

*then $z$ is an approximate root of $f$ associated with $c^{\frac{1}{d}}$. Otherwise if $0 < |c| < 1$ and $z$ satisfies*

$$|z - c^{\frac{1}{d}}| \leq \frac{3-\sqrt{7}}{d} |c| \leq \frac{3-\sqrt{7}}{d-1} |c^{\frac{1}{d}}|$$

*then $z$ is an approximate root of $f$ associated with $c^{\frac{1}{d}}$.*

This example becomes useful to us when we approximate a root of a univariate binomial.

## 2 An Algorithm for Univariate Binomials

In this section we present algorithms for approximating a root of a system of binomials. We begin with the question of a monic univariate binomial and extend our method to other systems of binomials.

## 2.1 Approximating a $d$th Root of a Positive Number

Since complex numbers can be written in polar form with some real, nonnegative radius, it is useful to consider methods for approximating the $d$th root of some positive number. We use the bisection method, which we formalize as follows:

**Bisection Algorithm.** *Input: A positive number $c$ and the desired exponent $d$. Let $\varepsilon$ be a lower bound on $\frac{3-\sqrt{7}}{2\gamma(f,c^{\frac{1}{d}})}$.*

1. *If $c = 1$ return 1. Else, if $c > 1$ set $a = 0$ and $b = c$. Otherwise set $a = 0$ and $b = 1$.*
2. *Compute $mid = \frac{b-a}{2}$.*
3. *Evaluate $f$ at $mid$.*
4. *If $f(mid) = 0$ or the algorithm has been applied $\log_2(c) - \log_2(\varepsilon)$ (resp. $-\log_2(\varepsilon)$) times, return $mid$. Else if $sgn(f(mid)) = sgn(f(a))$ set $a = mid$ and repeat steps 2-4. Otherwise set $b = mid$ and repeat steps 2-4.*

**sgn is the sign function*

The algorithm can be implemented fairly simply. We include some sage code below:

```
def approxAux(f, a, b, n, N):
    mid = (float(a)+float(b))/float(2)
    if (sign(f(mid))==0) or (n>=N):
        return mid
    elif sign(f(mid))==sign(f(a)):
        return approxAux(f,mid,b,(n+1),N)
    else:
return approxAux(f,a,mid,(n+1),N)

def bisectionApprox(f, a, b, eps):
    n = 1
    N = (logInt((b-a),2))-logInt(eps,2)
    return approxAux(f,a,b,n,N)
```

Note that in order to count the number of iterations we must compute $\log_2(c) - \log_2(\varepsilon)$. However, a fairly simple algorithm which tests repeated powers of two that runs in time $\log_2(a)$ can be used to approximate $\lceil \log_2(a) \rceil$ for real number $a$. We exclude this process from our discussion for the sake of simplicity. We would like to use the bisection method in approximating $d$th roots of a complex number. First we give the following lemma:

**Lemma.** *A root of a random binomial of the form $f(x_1) := x_1^d - c$ for $c > 0$ and $d \in \mathbb{N}$ can be approximated to within $\varepsilon$ in time $O(\log(\varepsilon)\log(d)^2)$ on average using the bisection method.*

*Proof.* After $k$ iterations of the bisection algorithm we have $|mid - \zeta| \leq \frac{c}{2^k}$ (resp. $\frac{1}{2^k}$. Thus applying the bisection algorithm $\log_2(c) - \log_2(\varepsilon)$ (resp. $-\log_2(\varepsilon)$) times will return $mid$ such that

$$|mid - \zeta| \leq \frac{c\varepsilon}{c} \leq \frac{3-\sqrt{7}}{2\gamma(f, c^{\frac{1}{d}})}$$

and similarly for the case where $0 < |c| < 1$. The complexity of evaluating $x_1^d - c$ at each iteration is $O(\log(d)^2)$ on average. Thus the overall average complexity of approximating $c^{\frac{1}{d}}$ is $O(\log(\varepsilon)\log(d)^2)$. □

## 2.2 The Algorithm

We present a method for approximating the roots of a univariate binomial with complex coefficients. The key observation is that for a complex number $c$ written in polar form as $re^{i\theta}$, we have that $c^{\frac{1}{d}} = r^{\frac{1}{d}} e^{\frac{i\theta}{d}}$.

**Univariate Binomial Algorithm.** *Input: A univariate polynomial $f(x_1) := c_1 + c_2 x_1^d$.*

1. *Let $c = \frac{c_1}{c_2}$. If $|c| \geq 1$ let $\varepsilon = \frac{1}{3d}$. Otherwise if $0 < |c| < 1$ let $\varepsilon = \frac{3-\sqrt{7}}{2}|c|$.*

2. *Let $r = |c|$ and approximate $r^{\frac{1}{d}}$ to within $\frac{\varepsilon}{5}$ using bisection. Call this approximation $r_0$.*

3. *If $c$ is written $a + bi$, the argument of $c$ can be computed in terms of $\arctan(\frac{b}{a})$. Approximate the argument of $c$, $\theta$, by approximating $\arctan\left(\frac{b}{a}\right)$ with Taylor series. Truncate the Taylor series after $\frac{\frac{5}{d\varepsilon}-3}{2r}$ terms if $r \geq 1$ and after $\frac{\frac{5}{d\varepsilon}-3}{2}$ terms otherwise. Call this approximation $\alpha$. Note that if the coefficients are entered in polar form this step is not necessary.*

4. *If $|r| \geq 1$ let $k = \max\{\frac{\lceil \log_{\frac{\pi}{2}}(\frac{\varepsilon}{5r})\rceil}{2} - 2, \frac{\lceil \log_{\frac{1}{2}}(\frac{\varepsilon 8!}{5r 8^8 2^8})\rceil + 8}{2}\}$, to ensure that $k$ is positive. Otherwise let $k = \max\{\frac{\lceil \log_{\frac{\pi}{2}}(\frac{\varepsilon}{5})\rceil}{2} - 2, \frac{\lceil \log_{\frac{1}{2}}(\frac{\varepsilon 8!}{5 8^8 2^8})\rceil + 8}{2}\}$.*

5. *Approximate $e^{i\frac{\alpha}{d}}$ via Taylor series. Truncate the Taylor series after $k$ terms and call the approximations for the real and imaginary components $s_k$ and $t_k$ respectively.*

6. *Return $r_0(s_k + it_k)$.*

An implementation of this algorithm is below:

```
def monicBinomApprox(c1,c2,d):
    c = c1*(c2^(-1))
    def binom(x):
     return x^d+c
```

```
   r = abs(c)
   if r > 1:
      eps = 1.0/float(15*d*r)
      alph = argTaylor(-c,1.0/float(15*r))
      r0 = bisectionApprox(binom, 0, r, eps)
      sk = cos(float(alph)/float(d), eps/r0)
      tk = sin(float(alph)/float(d), eps/r0)
      return r0*(sk+I*tk)
   elif r == 1:
     eps = 1.0/float(15*d*r)
     r0 = 1
alph = argTaylor(-c,1.0/float(15*r))
sk = cos(float(alph)/float(d), eps)
tk = sin(float(alph)/float(d), eps)
return r0*(sk+I*tk)
   else:
      eps = float((3-sqrt(7))*r)/float(5*d)
      alph = argTaylor(-c,float((3-sqrt(7))*r)/5.0)
      r0 = bisectionApprox(binom, 0, 1, eps)
      sk = cos(float(alph)/float(d), eps)
      tk = sin(float(alph)/float(d), eps)
      return r0*(sk+I*tk)
```

Our implementation makes use of auxiliary functions “argTaylor”,“sin”, and “cos” which compute the Taylor series approximations of $\arctan(\frac{b}{a})$, $\sin(\frac{\alpha}{d})$ and $\cos(\frac{\alpha}{d})$. We now prove that our algorithm returns an approximate root of $f$.

*Proof.* We begin with the case where $|c| \geq 1$. By the triangle inequality and the Mean Value Theorem we have that

$$|r^{\frac{1}{d}}e^{\frac{i\theta}{d}} - r_0(s_k + it_k)|$$

$$\leq r^{\frac{1}{d}}\left|\cos\left(\frac{\theta}{d}\right) - \cos\left(\frac{\alpha}{d}\right)\right| + r^{\frac{1}{d}}\left|i\sin\left(\frac{\theta}{d}\right) - i\sin\left(\frac{\alpha}{d}\right)\right| + \left|r^{\frac{1}{d}} - r_0\right| + r_0\left|\cos\left(\frac{\alpha}{d}\right) - s_k\right| + r_0\left|i\sin\left(\frac{\alpha}{d}\right) - it_k\right|$$

$$\leq r^{\frac{1}{d}}|\frac{\theta}{d} - \frac{\alpha}{d}| + r^{\frac{1}{d}}|\frac{\theta}{d} - \frac{\alpha}{d}| + \frac{\varepsilon}{5} + \frac{|\alpha|^{2k+2}}{(2k+2)!} + \frac{|\alpha|^{2k+3}}{(2k+3)!}$$

$$\leq \frac{1}{d}\frac{1}{2\left(\frac{\frac{5}{d\varepsilon}-3}{2}\right)+3} + \frac{1}{d}\frac{1}{2\left(\frac{\frac{5}{d\varepsilon}-3}{2}\right)+3} + \frac{\varepsilon}{5} + \frac{\pi^{2k+2}}{(2k+2)!} + \frac{\pi^{2k+3}}{(2k+3)!}$$

$$\leq \frac{\varepsilon}{5} + \frac{\varepsilon}{5} + \frac{\varepsilon}{5} + \frac{\pi^{2k+2}}{(2k+2)!} + \frac{\pi^{2k+3}}{(2k+3)!}$$

Because $2k+3 > 2k+2 \geq 2 > \frac{\pi}{2}$, the sequence $\{\frac{\pi^j}{j!}\}$ is strictly decreasing for $j \geq \frac{\pi}{2}$, and $2^j \leq j!$ for $j \geq 2$, we have

$$\frac{\pi^{2k+3}}{(2k+3)!} \leq \frac{\pi^{2k+2}}{(2k+2)!} \leq \left(\frac{\pi}{2}\right)^{2k+2} \leq \frac{\varepsilon}{5}$$

Thus we have

$$|r^{\frac{1}{d}} e^{\frac{i\theta}{d}} - r_0(s_k + it_k)| \leq \frac{\varepsilon}{5} + \frac{\varepsilon}{5} + \frac{\varepsilon}{5} + \frac{\varepsilon}{5} + \frac{\varepsilon}{5} = \varepsilon$$

□

## 2.3 Average-Case Complexity

We now examine the computational complexity of our algorithm for approximating a root of a univariate binomial.

**Proposition.** *The algorithm for approximating a root of a univariate binomial to within a given $\varepsilon$ has computational complexity $O(\log(\varepsilon)\log(d)^2 + \log(d) + \log(\varepsilon) + \log(\varepsilon)^3(\log(\log(\varepsilon))^2)$. In particular, the algorithm has average case complexity $O((\log d)^3(\log\log d)^2)$.*

*Proof.* The complexity of approximating the $d$th root of $|c|$ to within $\frac{\varepsilon}{5}$ is $O(\log(\varepsilon)\log(d)^2)$. Computing the value of the Taylor series approximation for the argument of $c$ has computational complexity $O(\log(\varepsilon)^2 + \log(d)^2)$. Computing the Taylor series approximation for $e^{\frac{i\alpha}{d}}$ has computational complexity $O(\log(\varepsilon)^3 \log\log(\varepsilon)^2 + \log(\varepsilon)\log\log(\varepsilon)$: $O(\log(\varepsilon)^3 \log\log(\varepsilon)^2)$ bit operations to compute the denominators and $O(\log(\varepsilon)\log\log(\varepsilon)$ to compute the numerators of the summands. For sufficiently large $\varepsilon$,

$$\log(\varepsilon)^3 \log\log(\varepsilon)^2 + \log(\varepsilon)\log\log(\varepsilon) \leq 2\log(\varepsilon)^3 \log\log(\varepsilon)^2$$

Thus the overall computational complexity of approximating a $d$th root of $c$ to within $\varepsilon$ is $O(\log(\varepsilon)\log(d)^2 + \log(d) + \log(\varepsilon) + \log(\varepsilon)^3(\log(\log(\varepsilon))^2)$.

To return an approximate root according to Smale's definition, our $\varepsilon = O(d)$. Thus the average case complexity of our algorithm is $O(\log(d)^3 + \log(d) + (\log d)^3(\log\log d)^2) = O((\log d)^3(\log\log d)^2)$. □

Note that we can also use this approximation to find a non-zero root of a binomial $c_1x_1^{a_1} + c_2x_1^{a_2}$: Assume WLOG that $a_1 < a_2$ and and apply the algorithm for approximating a root of a univariate binomial to the binomial $f(x_1) := c_1 + c_2x_1^{a_2-a_1}$, except instead of approximating a root to within $\frac{1}{3(a_2-a_1)}$ (or $\frac{3-\sqrt{7}}{(a_2-a_1)}\left|\frac{c_1}{c_2}\right|$) we approximate a root to within $\frac{1}{3a_2}$ (or $\frac{3-\sqrt{7}}{a_2}\left|\frac{c_1}{c_2}\right|$). The ability of this algorithm to approximate a nonzero root of a univariate binomial with a root at zero will become useful when we look at certain types of trinomials.

## 3 An Algorithm for Multivariate Binomial Systems

We have thus far considered univariate binomials. In this section we present algorithms for approximating the roots of $n$-variate systems of binomials. We first introduce some machinery which will become useful when examining this case.

### 3.1 Matrix Exponentiation and the Smith Normal Form

For a vector $x = (x_1, \ldots, x_n) \in \mathbb{C}^n$ and an $n \times n$ matrix of integers $A = \{a_{ij}\}$ define

$$x^A := (x_1^{a_{11}} \cdots x_n^{a_{n1}}, \ldots, x_1^{a_{1n}} \cdots x_n^{a_{nn}})$$

This notation relates matrices and systems of binomials. For instance, for a system of binomials

$$f(x) := \left\{ \begin{array}{ccc} x^{a_1} & - & c_1 \\ \vdots & & \vdots \\ x^{a_n} & - & c_n \end{array} \right. = \left\{ \begin{array}{ccc} x_1^{a_{11}} x_2^{a_{12}} \cdots x_n^{a_{1n}} & - & c_1 \\ \vdots & & \vdots \\ x_1^{a_{n1}} x_2^{a_{n2}} \cdots x_n^{a_{nn}} & - & c_n \end{array} \right.$$

we can instead write $f(x) := x^A - c$ where $c = (c_1, \ldots, c_n)$. Relating matrices and systems of binomials in this way allows us to use some of the techniques performed on matrices when approximating roots of a system of binomials.

**Definition – Smith Normal Form.** *We say that an $n \times n$ integer matrix $S$ is in Smith Normal Form if*

1. *It is diagonal*
2. *Its entries are positive*
3. *If* $S = \begin{bmatrix} d_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & d_n \end{bmatrix}$ *then* $d_i \mid d_{i+1} \forall i \in \{1, \ldots, n\}$

**Proposition.** *For any $n \times n$ matrix of integers $A$ there exists a unique $S$ such that $UAV = S$ for some $U, V \in SL(n, \mathbb{Z})$.*

**Theorem (Kannan and Bachem [1979]).** *There exists an algorithm which returns the Smith Normal Form of a given nonsingular $n \times n$ matrix $A$ and the multipliers $U$ and $V$ and runs in time polynomial in $n$ and* $\max |a_{ij}|$ *where $A = \{a_{ij}\}$.*

Kannan and Bachem also present such an algorithm explicitly. We use their algorithm to convert general systems of binomials to diagonal binomial systems.

## 3.2 The Algorithm

Approximating a root of an $n$-variate system of binomials $f$ requires computing an upper bound for $\frac{3-\sqrt{7}}{2\gamma(f,z)}$, now with $z \in \mathbb{C}^n$. We have that

**Lemma (Phillipson [2014]).** *For a diagonal system of binomials $f : \mathbb{C}^n \to \mathbb{C}^n$ and $x = (x_1, \dots, x_n)$ we have that*

$$\gamma(f,x) \leq \frac{\sqrt{2n}X \max\{|x_1^{-a_i}|\}||x||_1^{d-2}d^2}{2}$$

*where $X = \max\{|x_i|\}$ and $||x||_1 = \sqrt{1+\sum_{i=1}^n |x_i|^2}$*

Thus for a true root $\zeta = (\zeta_1, \dots, \zeta_n)$ we have that

$$\gamma(f,\zeta) \leq \frac{\sqrt{2n}||c||_\infty \max\{\frac{1}{|c_i|}\}||c||_1 d^2}{2}$$

if $||c|| > 1$ and

$$\gamma(f,\zeta) \leq \frac{\sqrt{2n}\max\{\frac{1}{|c_i|}\}d^2}{2}$$

otherwise.

**Algorithm for Diagonal Systems of Binomials.** *Input: a diagonal system of binomials*

$$\begin{cases} x_1^{a_1} & - & c_1 \\ \vdots & & \vdots \\ x_n^{a_n} & - & c_n \end{cases}$$

*where each $a_i \in \mathbb{Z}$.*

1. *Let $\varepsilon$ be $\frac{3-\sqrt{7}}{\sqrt{2n}||c||_\infty \max\{\frac{1}{|c_i|}\}||c||_1 d^2}$ if $||c|| > 1$ and $\frac{3-\sqrt{7}}{\sqrt{2n}\max\{\frac{1}{|c_i|}\}d^2}$ otherwise.*
2. *Approximate each $\zeta_i$ to within $\frac{\varepsilon}{\sqrt{n}}$ using our algorithm for approximating the roots of a univariate binomial and call the approximation $\alpha_i$.*
3. *return $\alpha = (\alpha_1, \dots, \alpha_n)$.*

We show that our algorithm does indeed return an approximate root of $f$:

*Proof.*

$$||\alpha - \zeta|| \leq \sqrt{n}\max_i\{|\alpha_i - \zeta_i|\} \frac{\sqrt{n}\varepsilon}{\sqrt{n}} = \varepsilon$$

$\square$

### 3.3 Average-Case Complexity

**Proposition.** *The average case complexity of the algorithm for diagonal systems of binomials is* $O((\log(d)+\log(n))\log(d)^2 n + (\log(d)+\log(n))^3 \log(\log(d)+\log(n))n)$.

*Proof.* Recall that the computational complexity of our algorithm for approximating a root of a univariate polynomial to within $\varepsilon$ is $O(\log(\varepsilon)\log(d)^2+\log(d)+\log(\varepsilon)+\log(\varepsilon)^3(\log(\log(\varepsilon))^2)$. For us, $\varepsilon = O(log(d)+log(n)$. Plugging in this value for $\varepsilon$ reduces to $O((\log(d)+\log(n))\log(d)^2 n + (\log(d)+\log(n))^3\log(\log(d)+\log(n))n)$. $\square$

We hope to use the Smith Normal Form and Kannan and Bachem's algorithm to reduce more general systems of binomials to the case of a diagonal system of binomials. In the future we hope to provide a sufficiently small upper bound on $\gamma(f,x)$ for a general system of binomials $f$ so that our algorithm will run quickly.

## 4 A Partial Algorithm for Univariate Trinomials

In this section we present an algorithm for approximating a root of a certain kind of univariate trinomial. We do this by reducing the trinomial to its "lower binomials" and applying some results from $\alpha$-theory.

### 4.1 Some Background

**Definition – Convex Hull.** *Let $S$ be a finite subset of $\mathbb{R}^n$. The convex hull of $S$ is*

$$Conv(S) := \{\textstyle\sum_{i=1}^t \alpha_i x_i | \alpha_i \geq 0, \sum_{i=1}^t \alpha_i = 1\}.$$

**Definition – Archimedean Newton Polytope.** *For $f(x_1) := c_1 x_1^{a_1} + \ldots + c_t x_1^{a_t} \in \mathbb{C}[x_1]\setminus\{0\}$ we define*

$$ArchNewt(f) := Conv(\{(a_i, -\log|c_i|) | i \in \{1,\ldots,t\}\}).$$

**Definition – Lower Polynomial.** *For $f(x_1) := c_1 x_1^{a_1} + \ldots + c_t x_1^{a_t} \in \mathbb{C}[x_1]\setminus\{0\}$ with $0 < a_1 < \ldots < a_t$ we define $g(x_1) := \sum_{i\in I} c_i x_1^{a_i}$ to be a lower polynomial of $f$ if and only if there exists $w \in \mathbb{R}$ such that $\{(a_i, -\log|c_i|) | i \in I\}$ is exactly the intersection of $\{(a_i, -\log|c_i|) | i \in \{1,\ldots,t\}\}$ with the face of ArchNewt$(f)$ that has outer normal $(w,-1)$.*

In particular, for a univariate trinomial $f(x_1) := 1 + cx^d \pm x^D$ we have that the lower polynomials of $f$ are

- $1 \pm x^D$ if $|c| < 1$
- $f$ if $|c| = 1$
- $1 + cx^d$ and $cx^d \pm x^D$ if $|c| > 1$

Because we already have an algorithm for approximating a root of a binomial, we would like to reduce the case of trinomials to a question of approximating a root of a binomial. Some results of Avendaño will be useful:

**Definition – $W$-property (Avendaño [2008]).** $f(x_1) := c_1x_1^{a_1} + \ldots + c_tx_1^{a_t} \in \mathbb{C}[x_1]$. *We say $f$ has the $W$-property iff the following implication holds: $(a_i, -\log|c_i|)$ is within vertical distance $W$ of the lower hull of ArchNewt($f$) $\Longrightarrow$ $(a_i, -\log|c_i|)$ is a lower vertex of ArchNewt($f$).*

**Proposition (Avendaño [2008]).** *Let $f(x_1) := c_1x_1^{a_1} + \ldots + c_tx_1^{a_t} \in \mathbb{C}[x_1]$. If $f$ satisfies the $W$-property with $W \geq \max\{\log_2(2td), \log_2(4t^2d^2/0.03\}$ (where $d = \max\{a_i\}$) then any nonzero root $x$ of a lower polynomial of $f$ satisfies $\alpha(f,x) < 0.03$.*

**Robust $\alpha$-Theorem (Blum et al. [1998]).** *There is a positive real number $u_0 \geq 0.05$ such that if $\alpha(f,z) < 0.03$, then there is a root $\zeta$ of $f$ such that*

$$B\left(\frac{u_0}{\gamma(f,z)}, z\right) \subset B\left(\frac{3-\sqrt{7}}{2\gamma(f,\zeta)}, \zeta\right)$$

## 4.2 The Algorithm

**Partial Algorithm for Trinomials.** *Input: A univariate trinomial $f(x_1) := c_1 + c_2x_1^d + c_3x_1^D$.*

1. *If $d = 1$ and $D = 2$ use the quadratic formula to solve for the roots of $f$.*
2. *If $c_1 \neq 1$ or $c_3 \neq \pm 1$ then define the following constants: $\mu = \frac{1}{c_1}$, $\rho = \left(\frac{c_1}{c_3}\right)^{\frac{1}{D}}$, and $\nu = \frac{c_2}{c_1}\left(\frac{c_1}{c_3}\right)^{\frac{d}{D}}$.*
3. *Define $g(x_1) = f(\rho x_1)$, so that $\mu g(x_1) = 1 + \nu x_1^d \pm x_1^D$.*
4. *If $\mu g$ has the $W$-property, we perform steps 4-6: use the algorithm for monic univariate binomials to approximate a root of the lower binomial of degree $D$ to within $\frac{\varepsilon}{(3-\sqrt{7})20}$, where $\varepsilon$ is as in the univariate binomial case. Call this approximation $\alpha_0$.*
5. *Approximate $\rho = \left(\frac{c_1}{c_3}\right)^{\frac{1}{D}}$ to within $\frac{\varepsilon}{(3-\sqrt{7})20|\frac{c_1}{c_3}|}$ if $\left|\frac{c_1}{c_3}\right| \geq 1$ and to within $\frac{\varepsilon}{(3-\sqrt{7})20}$ otherwise, and call the approximation $p$.*
6. *Let $\alpha = p\alpha_0$ and return $z$.*

We now show that this algorithm returns an approximate root of $f$.

*Proof.* Observe that

$$\begin{aligned}\mu f(\rho x_1) &= \mu c_1 + \mu c_2 \rho^d x_1^d + \mu c_3 \rho^D x_1^D x \\ &= \left(\frac{1}{c_1}\right) c_1 + \frac{c_2}{c_1}\left(\frac{c_1}{c_3}\right)^{\frac{d}{D}} x_1^d + \frac{c_3}{c_1}\left(\frac{c_1}{c_3}\right) x_1^D \\ &= 1 + \nu x_1^d \pm x^D\end{aligned}$$

How do we use this reduction to approximate the roots of $f$? Define $g(x_1) := f(\rho x_1)$, so that $\mu g(x_1) = 1 + \nu x_1^d \pm x_1^D$. Note that

$$\gamma(\mu g, z) = \sup_{k\geq 2}\left|\frac{\mu g^{(k)}(z)}{\mu g'(z)k!}\right|^{\frac{1}{k-1}} = \sup_{k\geq 2}\left|\frac{g^{(k)}(z)}{g'(z)k!}\right|^{\frac{1}{k-1}} = \gamma(g,z)$$

and

$$\beta(\mu g, z) = \left|\frac{\mu g(z)}{\mu g'(z)}\right| = \left|\frac{g(z)}{g'(z)}\right| = \beta(g,z)$$

Thus $\alpha(\mu g, z) = \alpha(g, z)$. Also, by Avendaño we have that $\alpha(g, z) = \alpha(f, \rho z)$. We also know that if $\zeta$ is a true nonzero root of a lower binomial of $\mu g$ the $\alpha(\mu g, \zeta) < \alpha_0$. Thus we have that if $\zeta$ is a true nonzero root of the lower binomial of degree $D$ of $\mu g$, then $\alpha(f, \rho\zeta) < \alpha_0$. All that remains to be done is to pick some $z$ within $\frac{u_0}{\gamma(f,\rho\zeta)}$ of $\rho\zeta$. By Avendaño we have that $|\rho|\gamma(f, \rho\zeta) = \gamma(g, \zeta) \leq \left|\frac{D-1}{2\zeta}\right|$, so $\gamma(f, \rho\zeta) \leq \left|\frac{D-1}{2\zeta|\rho|}\right|$. If we pick $\alpha$ according to our algorithm then we have that

$$||\alpha - \rho\zeta|| \leq |p|||\alpha_0 - \zeta|| + |p - \rho| \leq \min\left\{\frac{\varepsilon}{(3-\sqrt{7})10}, \frac{\varepsilon}{(3-\sqrt{7})10\left|\frac{c_1}{c_3}\right|}\right\} \leq \frac{\varepsilon}{(3-\sqrt{7})10|\rho|} \leq \frac{u_0}{\gamma(f,\rho\zeta)}$$

Then by the Robust $\alpha$ Theorem $z$ is an approximate root of $f$. □

## 4.3 Computational Complexity

**Proposition.** *The average case complexity of approximating a root of a univariate trinomial using our algorithm for univariate trinomials is* $O((\log D)^3(\log\log D)^2)$.

*Proof.* If $f$ is a quadratic, then computing a root of $f$ takes a constant number of bit operations on average, since on average each coefficient of $f$ is a constant. Otherwise we claim that $f$ satisfies the $W$-property in the average case. When $f$ satisfies the $W$-property approximating a root is simply a matter of performing the algorithm for univariate binomials twice, with $\varepsilon$ rescaled by a constant. Thus the computational complexity o the algorithm is unchanged. We claim that on average $f$ satisfies the $W$-property when it is not a quadratic, so the average case computational complexity of the algorithm is $O((\log D)^3(\log\log D)^2)$. □

### 4.4 A Discussion of Probability

In our analysis of the complexity of approximating a root of a univariate trinomial $f$ with our algorithm we claimed that on average $f$ satisfies the $W$-property. We discuss this assertion here. Smale does not specify a particular random variable for the coefficients of a random polynomial, so we use complex Gaussians with variances specified in this manner: If $f = c_1x_1^{a_1} + \ldots + c_tx_1^{a_t}$ where $0 < a_1 < \ldots < a_t$ we let $c_i$ be chosen from a complex normal distribution with variance $\binom{a_t}{a_i}$. In particular, for a trinomial $1 + \nu x_1^d \pm x^D$, $\nu$ is chosen from a complex normal distribution with variance $\binom{D}{d}$. Let $r$ denote the expected value of $|\nu|$. Then

$$r = \frac{\sqrt{\pi}}{4}\sigma^3$$

$d < D$ gives us that $\binom{D}{d}^3 \geq \frac{4}{\sqrt{\pi}}$ so the expected value of $|\nu|$ is greater than 1. Thus on average $(d, -\log|\nu|)$ is a vertex of the lower hull of ArchNewt$(f)$ and so $f$ satisfies the $W$-property.

## 5 Conclusion

We hope that the algorithms in this paper become useful for more elaborate systems of polynomials. In particular, we plan to construct algorithms which run quickly and approximate a root of a univariate trinomial which does *not* satisfy the $W$-property, as well as extend our results to $n$-variate systems of trinomials. We also wish to improve our algorithm for approximating roots of a system of binomials so that it runs in time polynomial in only $\log(d)$ and $\log(n)$. Our methods also will hopefully provide us with some tools to approach the questions of approximating a real root of a polynomial system, or the nearest root of a system closest to some query point.

## References

Martín Avendaño. Unpublished notes, 2008.

Lenore Blum, Felipe Cucker, Mike Shub, and Steve Smale. *Complexity and Real Computation.* Springer-Verlag, 1998.

Ravindran Kannan and Achim Bachem. "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix". *SIAM Journal on Computing*, 8, 1979.

Kaitlyn Phillipson. Unpublished notes, 2014.

Steve Smale. "Newton's Method Estimates from Data at One Point". In *The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics*, 1986.