Approximate polynomial gcd: small degree and small height perturbations

Joachim von zur Gathen

B-IT, Universität Bonn
53113 Bonn, Germany

Maurice Mignotte

Département de Mathématiques, Université Louis Pasteur
7 rue René Descartes, F-67084 Strasbourg cedex, France

Igor E. Shparlinski

Department of Computing, Macquarie University
NSW 2109, Australia

Abstract

We consider the following computational problem: we are given two coprime univariate polynomials f_0 and f_1 over a ring $\mathcal{R}$ and want to find whether after a small perturbation we can achieve a large gcd. We solve this problem for two notions of "large" (and "small"): large degree (when $\mathcal{R} = \mathbb{F}$ is an arbitrary field, in the generic case when f_0 and f_1 have a so-called normal degree sequence), and large height (when $\mathcal{R} = \mathbb{Z}$). The algorithms use polynomial time for the degree notion, and also for the height notion when the degree is fixed [1]. Our work adds to the existing notions of "approximate gcd".

Key words: Euclidean algorithm, gcd, approximate computation

Email addresses: gathen@bit.uni-bonn.de (Joachim von zur Gathen), mignotte@math.u-strasbg.fr (Maurice Mignotte), igor@ics.mq.edu.au (Igor E. Shparlinski)

[1] The latter condition is erroneously left out in the published version.

1. Introduction

Symbolic (exact) computations of the gcd of two univariate polynomials form a well-developed topic of computer algebra. These methods are not directly applicable when the coefficients are "inexact" real numbers, maybe coming from physical measurements, since then the gcd is almost always 1. The appropriate model here is to ask for a "large" gcd, allowing "small" additive perturbations of the inputs. Numerical analysis provides several ways of formalizing this precisely, and "approximate gcd" computations are an emerging topic of computer algebra with a growing literature. We only point out the pioneering work of Schönhage (1985), and Bini & Boito (2007); Emiris, Galligo & Lombardi (1997); Karcanias, Fatouros, Mitrouli & Halikias (2006); Karmarkar & Lakshman (1998); Li, Yang & Zhi (2005); Pan (2001); Rupprecht (1999) and the references therein.

The present paper introduces two "exact" notions of approximate gcds, where we allow "small" additive perturbations of the inputs and ask for a "large" . In the first setting we let $f_0, f_1 \in \mathbb{F}[x]$ be two univariate polynomials over a field $\mathbb{F}$, both of degree at most n and with a normal degree sequence in the Euclidean algorithm, and d and e integers. We are interested in perturbations $u_0, u_1 \in \mathbb{F}[x]$ of degree at most e such that $\deg \gcd(f_0+u_0, f_1+u_1) \geq d$. We show that if $e < \min\{2d-n, n-d\}$, then the problem has at most one solution, and if one exists, we can find it in polynomial time. In the second setting, we consider polynomials over $\mathbb{Z}$ and obtain an efficient algorithm for perturbations $u_1 \in \mathbb{Z}[x]$ of small height that achieve a $\gcd(f_0, f_1+u_1)$ of large height (without any restrictions on the degree except for $\deg u_1 \leq n$).

The latter result is based on the work of Howgrave-Graham (2001) on an analogous question for integers, and in fact can be viewed as an extension to polynomials of those results.

We prove that our algorithms solve our problem under rather restrictive assumptions. Several open questions are mentioned in Section 4. One of them is whether either a variant or some other algorithm can tackle a larger set of input values and provide a more practical solution. Finding multidimensional analogues, that is, constructing algorithms to find "small" perturbations $u_0, \ldots, u_{s-1}$ of $f_0, \ldots, f_{s-1}$ such that $\gcd(f_0+u_0, \ldots, f_{s-1}+u_{s-1})$ is "large" (in both number and polynomial cases) is another interesting direction of research.

Our approaches are quite different in spirit from the numerical ones, and we see no meaningful way of comparing them.

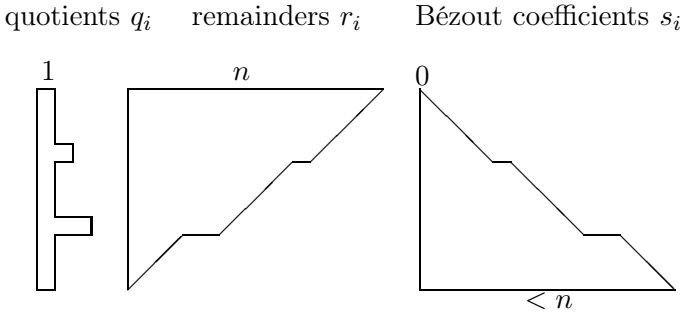

Figure 2.1: The degrees of the quotients (left), remainders (center), and Bézout coefficients in the EEA, starting at the top.

2. The degree measure

We write $f \text{ quo } g$ and $f \text{ rem } g$ for the quotient and remainder on division of f by nonzero g. Thus $f = (f \text{ quo} g)\cdot g + (f \text{ rem} g)$ and $\deg(f \text{ rem} g) < \deg g$.

The *degree sequence* of two univariate polynomials $f_0, f_1 \in \mathbb{F}[x]$ is the sequence of degrees $\deg f_0, \deg f_1, \deg f_2, \ldots$ of the remainders $f_0, f_1, f_2, \ldots$ in the Euclidean algorithm. Usually, but not always, $\deg f_{i-1} = 1 + \deg f_i$, and we say that f_0, f_1 *have a normal degree sequence* if that is the case for all i. We denote by M a polynomial multiplication time over $\mathbb{F}$, so that two polynomials of degree at most n can be multiplied with $O(\mathsf{M}(n))$ operations in $\mathbb{F}$. We may use $\mathsf{M}(n) = n \log n \log\log n$. In particular $\mathsf{M}(n) \in \mathcal{O}^{\sim}(n)$, where as usual $A \in \mathcal{O}^{\sim}(B)$ means that $|A| \le c_1 B(\log(B+2))^{c_2}$ for some constants $c_1, c_2 > 0$; see (von zur Gathen & Gerhard, 2003, Chapter 8).

For our first result, we consider a field $\mathbb{F}$ and univariate polynomials $f_0, f_1 \in \mathbb{F}[x]$. We ask for perturbations $u_0, u_1 \in \mathbb{F}[x]$ of small degree so that the perturbed polynomials have a gcd of large degree. More precisely, we also have integers e_0, e_1, d, and we consider the set

$$\begin{aligned}\mathcal{U} = \{(u_0, u_1) \in \mathbb{F}[x]^2 \colon \deg u_i \le e_i \ \text{ for } i = 0, 1, \\ \deg \gcd(f_0 + u_0, f_1 + u_1) = d\}.\end{aligned} \tag{2.1}$$

If e_i is negative, then the condition is meant to imply that $u_i = 0$. As an example, we can take $f_1, g, u_0 \in \mathbb{F}[x]$ of degrees n_1, m, e_0, respectively, with $e_0 < n_1 < m$, and $f_0 = gf_1 - u_0, d = n_1$, and $e_1 = n_1 - m - 1 < 0$. Then $\mathcal{U} = \{(u_0, 0)\}$, and the hypotheses in Theorem 2.3 below are satisfied.

It is well-known that the first quotients in the Extended Euclidean Algorithm (EEA) depend only on the top coefficients of the two input polynomials. For our question, it means that the first quotients are identical for the

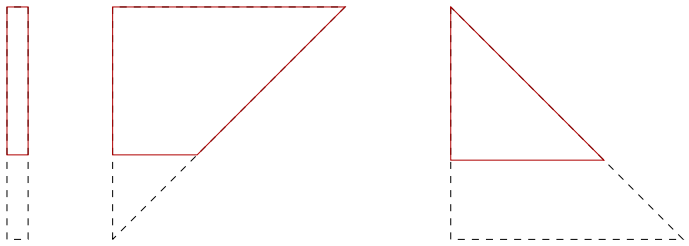

Figure 2.2: Figure 1 with a normal degree sequence and truncated bottom part to indicate a large gcd.

inputs and their (unknown) perturbations. Furthermore, the gcd is large if and only if the last quotients disappear.

The algorithm below executes the EEA for (f_0, f_1). It produces a finite series of "lines" (r_j, s_j, t_j) such that $s_j f_0 + t_j f_1 = r_j$, where $\deg r_j \le n$ is strictly decreasing with growing j (see von zur Gathen & Gerhard 2003, Section 3.2). We have $s_1 = t_0 = 0$, and all other s_i and t_i are nonzero. Furthermore, since $\deg s_j$ and $\deg t_j$ are strictly increasing (see von zur Gathen & Gerhard 2003, Lemma 3.10), there is at most one "line" (r, s, t) with a prescribed degree for s (or t). We denote as $lc(f)$ the leading coefficient of a polynomial f.

ALGORITHM 2.2. Approximate gcd of large degree.

Input: $f_0, f_1 \in \mathbb{F}[x]$ monic of degrees $n_0 > n_1$, respectively, coprime and with a normal degree sequence. Furthermore, integers d, e_0, e_1 with $d > 0$ and

$$e_0 < \min\{2d - n_1, n_0 - d\}, e_1 < \min\{2d - n_0, n_1 - d\}.$$

Output: $\mathcal{U}$ as in (2.1).

1. Execute the EEA with input $(f_0,\ f_1)$.
2. Check if the EEA computes (r, s, t) with $sf_0 + tf_1 = r$ and $n_0 - \deg t = n_1 - \deg s = d$. If not, return $\mathcal{U} = \varnothing$.
3. Otherwise, if $s = 0$, then let $u_0 = -(f_0 \text{ rem } f_1)$ and return $\mathcal{U} = \{(u_0, 0)\}$ if $\deg u_0 \le e_0$, and else $\mathcal{U} = \varnothing$. If $t = 0$, then return $\mathcal{U} = \varnothing$.
4. {We now have $sf_0 + tf_1 = r$ and $st \ne 0$.} Compute

$$h_0 = f_0 \text{ quo } t,$$
$$h_1 = f_1 \text{ quo } s.$$

If h_0 and h_1 are not associates, return $\mathcal{U} = \varnothing$.

5. Else, compute

$$\begin{aligned} h &= lc(h_0)^{-1} h_0, \\ \alpha &= lc(t)^{-1}, \\ q_0 &= \alpha t, \\ q_1 &= -\alpha s, \\ u_i &= q_i h - f_i \text{ for } i = 0, 1. \end{aligned}$$

6. If $\deg u_i \leq e_i$ for $i = 0, 1$, then return $\mathcal{U} = \{(u_0, u_1)\}$, else return $\mathcal{U} = \varnothing$.

THEOREM 2.3. *Let f_0, f_1, $n = n_0$, n_1, d, e_0, e_1 satisfy the input specification of Algorithm 2.2. Then the set $\mathcal{U}$ contains at most one element, and Algorithm 2.2 computes it with $O(\mathsf{M}(n) \log n)$ operations in $\mathbb{F}$.*

PROOF. We have noted above that there is at most one "line" (r, s, t) in the EEA with $sf_0 + tf_1 = r$ and $n_0 - \deg t = n_1 - \deg s = d$. If there is no such line, then our algorithm returns $\mathcal{U} = \varnothing$. Otherwise we take that line.

We first have to check that any (u_0, u_1) returned by the algorithm is actually in the set $\mathcal{U}$. This is clear in Step 3. For an output in Step 6, we note that

$$\gcd(f_0 + u_0, f_1 + u_1) = \gcd(q_0 h, q_1 h) = h \gcd(s, t) = h,$$

since $\gcd(s, t) = 1$ (see von zur Gathen & Gerhard 2003, Lemma 3.8 (v)),

$$\deg h = \deg h_0 = \deg f_0 - \deg t = d,$$

and indeed $(u_0, u_1) \in \mathcal{U}$.

To show the correctness of the algorithm it remains to show that if $\mathcal{U} \neq \varnothing$, then the algorithm does indeed return this set $\mathcal{U}$, and that $\mathcal{U}$ has at most one element.

So we now suppose that $\mathcal{U} \neq \varnothing$, let $(u_0, u_1) \in \mathcal{U}$, and $h = \gcd(f_0 + u_0, f_1 + u_1)$, so that $\deg h = d$. One first checks that the algorithm deals correctly with the two special cases $d = n_0$ and $d = n_1$. In the other cases, there exist uniquely determined $q_0, q_1 \in \mathbb{F}[x]$ such that

$$f_i = q_i h - u_i \quad \text{for} \quad i = 0, 1, \tag{2.4}$$

since $\deg u_i < 2d - n_{1-i} < d = \deg h$. Eliminating h from these two equations, we find

$$q_1 f_0 - q_0 f_1 = q_0 u_1 - q_1 u_0, \tag{2.5}$$

and call this polynomial $g = q_0 u_1 - q_1 u_0$. We have $\deg q_0 = n_0 - d < n_0$. Now g is nonzero, because otherwise f_0 would divide q_0, a polynomial of smaller degree than f_0, which would imply that $q_0 = 0$, a contradiction.

We have

$$\deg q_0 + \deg g \le n_0 - d + \max\{(n_0 - d) + e_1, (n_1 - d) + e_0\} < n_0,$$

since $e_i < 2d - n_{1-i}$ for $i = 0, 1$.

Thus (2.5) satisfies the degree inequalities of the EEA, and by the well-known uniqueness property of polynomial continued fractions (see, for example, von zur Gathen & Gerhard 2003, Lemma 5.15), there exist a remainder r and corresponding Bézout coefficients s, t in the EEA for f_0 and f_1, and nonzero $\alpha \in \mathbb{F}[x]$ such that

$$s f_0 + t f_1 = r \text{ and } (g, q_1, -q_0) = \alpha(r, s, t).$$

Furthermore, since the Euclidean degree sequence is normal, α is a constant. We have $n_0 - \deg q_0 = n_0 - \deg t = d$, similarly $n_1 - \deg q_1 = d$, and $\deg u_i \le e_i < n_i - d = \deg q_i$, so that u_i equals the remainder of f_i on division by q_i, for $i = 0, 1$. It follows from (2.4) that (u_0, u_1) is indeed returned by the algorithm.

In particular, since at most one (u_0, u_1) is returned by the algorithm and it equals each element of $\mathcal{U}$ (if $\mathcal{U} \ne \varnothing$), $\mathcal{U}$ contains at most one element.

The cost for computing a single line in the Extended Euclidean Scheme is $O(\mathsf{M}(n) \log n)$; see von zur Gathen & Gerhard 2003, Algorithm 11.4. All other operations are not more expensive. □

In particular the cost of Algorithm 2.2 is in $\mathcal{O}^{\sim}(n)$.

Figure 2.3 shows at the bottom the triangle of values in the e_0-d-plane satisfying the restriction required for e_0, with large $n_0 = n_1 + 1$. There are trivial solutions $u_i = -f_i \bmod h$ for $i = 0, 1$ when $e_0, e_1 \ge d - 1$, for any h of degree d; these form the area above the diagonal. We ran experiments with "random" polynomials, with and without a planted perturbed gcd. Values in the bottom triangle were, of course, correctly dealt with. We also ran the algorithm without any of the bounds d, e_0, e_1. Then it would typically compute $(u_0, u_1) \in \mathcal{U}$ with $e_0 = n_0 - d$ and $1 \le d \le n_1$, which is the dotted line in Figure 2.3. Planted gcds with $d < n_0/2$ were usually not detected.

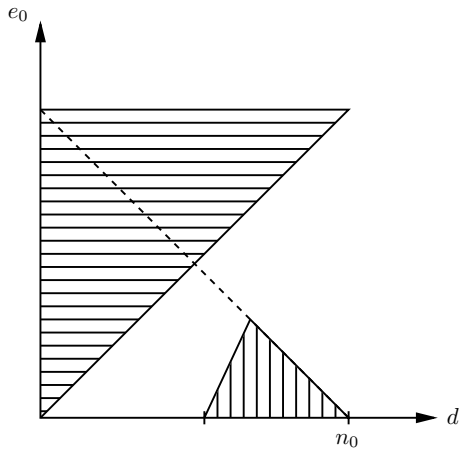

Figure 2.3: The three areas – bottom triangle, triangle above the diagonal, dotted line – are explained in the text.

3. The height measure

We now look at the same problem in a different setting which we consider only for polynomials over $\mathbb{Z}$ (although it can be extended to polynomials over other suitable fields and rings). Namely, we consider the case where the height $H(f) = \max\{|c_j| : 0 \le j \le n\}$ of a polynomial

$$f = \sum_{j=0}^{n} c_j x^j \in \mathbb{Z}[x]$$

is the measure of interest.

We first need to know that a large polynomial takes a small value only very rarely. It might come as a surprise that, according to the following precise version, the bound for points with small values is the same as the one for roots.

LEMMA 3.1. *Let $f \in \mathbb{Z}[x]$ be nonzero of degree n, let $A \ge 2$ be an integer, and*

$$\mathcal{A} = \left\{a \in \mathbb{Z} : \ -A \le a \le A, \ |f(a)| \le 2^{-n-1}(n-1)!A^{-n}H(f)\right\}.$$

Then $\#\mathcal{A} \le n$.

PROOF. Clearly we can assume that $A \geq n/2$ since otherwise there is nothing to prove. Let $-A \leq a_0 < \ldots < a_n \leq A$ be $n+1$ arbitrary distinct integers. If we define $f_i = f(a_i)$ for $i = 0, \ldots, n$, then Lagrange interpolation says that

$$f = \sum_{i=0}^{n} f_i L_i, \tag{3.2}$$

where

$$L_i = g_i(x)/g_i(a_i),$$
$$g_i = \prod_{\substack{j=1 \\ j \neq i}}^{n} (x - a_j),$$

for $0 \leq i \leq n$. Since $A \geq n/2$, the height of any g_i can be estimated as

$$H(g_i) \leq \max_{0 \leq k \leq n} \binom{n}{k} A^{n-k} = \max\{nA^{n-1}, A^n\} \leq 2A^n.$$

With $n_0 = \lfloor n/2 \rfloor$ we have for all $i \leq n$

$$|g_i(a_i)| \geq n_0!(n-n_0)! = \binom{n}{n_0}^{-1} n! \geq 2^{-n} n!\,.$$

We now see from (3.2)

$$\begin{aligned} H(f) &\leq (n+1) \max_{0 \leq i \leq n} |f_i| \frac{H(g_i)}{|g_i(a_i)|} \\ &\leq \frac{1}{n!} 2^{n+1} (n+1) A^n \max_{0 \leq i \leq n} |f_i| \leq \frac{1}{(n-1)!} 2^{n+1} A^n \max_{0 \leq i \leq n} |f_i|, \end{aligned}$$

which concludes the proof. □

We need the following statement which has essentially been shown by Howgrave-Graham (2001). For the sake of completeness we present a succinct proof. The gcd of two integers, at least one of which is nonzero, is taken to be positive.

LEMMA 3.3. *Let F_0 and F_1 be integers, with $F_0 \neq 0$. Then the set of all integers V with $|V| < |F_1|$ and*

$$\gcd(F_0, F_1 + V) \geq 2\sqrt{|F_0 V|}$$

can be computed in time polynomial in $\log(|F_0 F_1| + 1)$.

PROOF. We may assume that $F_1 \neq 0$. For an integer V in the set we write

$$\Delta = \gcd(F_0, F_1 + V), \qquad G_0 = \frac{F_0}{\Delta}, \qquad G_1 = \frac{F_1 + V}{\Delta}.$$

Then $|F_1 + V| < 2|F_1|$, and

$$\begin{aligned}\left|\frac{F_0}{F_1} - \frac{G_0}{G_1}\right| &= \frac{|F_0 G_1 - F_1 G_0|}{\Delta |F_1 G_1|} = \frac{|F_0 V|}{\Delta |F_1 G_1|} \\ &\leq \frac{\Delta^2}{4\Delta |F_1 G_1|} \leq \frac{\Delta}{2 \cdot |(F_1 + V) G_1|} = \frac{1}{2G_1^2}.\end{aligned}$$

Thus G_0/G_1 is one of the convergents in the continued fraction expansion of F_0/F_1, and can be found in polynomial time. Furthermore, $\Delta = F_0/G_0$ can take only polynomially many values. For each of them, we verify whether $V = G_1\Delta - F_1$ satisfies the condition of the lemma. □

The gcd of polynomials f_0 and f_1 in $\mathbb{Z}[x]$ is monic if one of f_0 or f_1 is. We now consider for given $f_0, f_1 \in \mathbb{Z}[x]$ and integers D, E the set

$$\mathcal{V} = \{v \in \mathbb{Z}[x] \,:\, \deg v \leq n,\ H(v) \leq E,\ H(\gcd(f_0, f_1 + v)) \geq D\}. \tag{3.4}$$

The idea for a solution is the following probabilistic approach. Given $\epsilon > 0$, we choose $n+1$ random integers a in $\{-A, \ldots, A\}$ for $A = \lceil 4\varepsilon^{-1} n(n+1) \rceil$. Suppose we have v as in (3.4), and let $h = \gcd(f_0, f_1 + v)$. Then $h(a)$ divides $\gcd(f_0(a), f_1(a) + v(a))$ for all a. In an appropriate sense, f_0, f_1, v, and a are small, so that also all values $f_0(a)$, $f_1(a)$, $v(a)$ are small. By Lemma 3.1, with probability at least $1 - \varepsilon$ all $h(a)$ are large. The conditions for Howgrave-Graham's integer result are satisfied, and his method finds efficiently the set of all $v(a)$. Trying all interpolation polynomials v solves our task.

ALGORITHM 3.5. Approximate gcd of large height.

Input: $f_0, f_1 \in \mathbb{F}[x]$ of degrees $n \geq n_1 \geq 1$, respectively, with f_0 monic and $\gcd(f_0, f_1) = 1$. Furthermore, we are given a positive $\varepsilon < 1$ and positive integers D and E.

Output: $\mathcal{V}$ as in (3.4) or "failure".

1. Initialize $\mathcal{V} = \varnothing$. Put $A = \lceil 4\varepsilon^{-1} n(n+1) \rceil$ and choose $n+1$ distinct integers $a_0, \ldots, a_n$ uniformly at random in the interval $\{-A, \ldots, A\}$.
2. Evaluate $f_i(a_j)$ and check whether
$$|f_i(a_j)| > 2^{-n-1}(n-1)!A^{-n}H(f_i)$$
for each $j = 0, \ldots, n$ and $i = 0, 1$. Return "failure" if the check fails.
3. For each $j = 0, \ldots, n$, compute continued fraction expansions of the fractions $f_0(a_j)/f_1(a_j)$ and find the set of all integers V_j with
$$|V_j| < |f_1(a_j)| \qquad \text{and} \qquad \gcd\left(f_0(a_j), f_1(a_j) + V_j\right) \geq D2^{-n}A^{-n^2}.$$
4. For each possible choice $(V_0, \ldots, V_n)$ compute the unique interpolation polynomial $v \in \mathbb{Q}[x]$ of degree at most n with $v(a_j) = V_j$ for all j. If v satisfies the conditions in (3.4), then add v to $\mathcal{V}$.
5. Return $\mathcal{V}$.

THEOREM 3.6. *Let f_0, f_1, ε, D, E be inputs to Algorithm 3.5 with*
$$E < H(f_1)2^{-n-2}(n-1)!(4\varepsilon^{-1}n(n+1)+1)^{-2n},$$
$$D \geq \frac{1}{(n-1)!}2^{n+3}(4\varepsilon^{-1}n(n+1)+1)^{2n}(H(f_0)E)^{1/2}.$$
Then Algorithm 3.5 returns "failure" in step 2 with probability at most ε, and otherwise computes $\mathcal{V}$. It uses time polynomial in $(\log(DH(f_1)\varepsilon^{-1}))^n$.

PROOF. Let $v \in \mathcal{V}$ as in (3.4), $h = \gcd(f_0, f_1 + v)$, $d = \deg h$, and $H_i = H(f_i)$ for $i = 0, 1$. We want to show that with probability at least $1 - \varepsilon$, the polynomial v is found in Step 4.

We have $A = \lceil 4\varepsilon^{-1}n(n+1) \rceil > n$ and hence $(d-1)!A^{-d} \geq (n-1)!A^{-n}$. For $a_0, \ldots, a_n$ chosen in Step 1, by Lemma 3.1 we see that with probability at least
$$\left(1 - \frac{4n}{2A+1}\right)^{n+1} > \left(1 - \frac{\varepsilon}{2(n+1)}\right)^{n+1} > 1 - \varepsilon,$$

we have simultaneously

$$|h(a_j)| \geq 2^{-d-1}(d-1)!A^{-d}H(h) \geq 2^{-n-1}(n-1)!A^{-n}D$$

and

$$|f_i(a_j)| > 2^{-n-1}(n-1)!A^{-n}H_i$$

for each $j = 0, \ldots, n$ and $i = 0, 1$, since each a_j has to avoid the at most $d + 2n \leq 3n$ "small" values of h, f_0 and f_1, and also the values $a_0, \ldots, a_{j-1}$. We also have

$$|f_1(a_j)| > 2^{-n-1}(n-1)!A^{-n}H_1 > 2A^nE \geq |v(a_j)|$$

for each j, so that $f_1(a_j) + v(a_j) \neq 0$. Now h is monic, so that the quotients f_0/h and $(f_1 + v)/h$ are integer polynomials. For any $a \in \mathbb{Z}$ it follows that $h(a)$ divides $\gcd(f_0(a), f_1(a))$. Thus we find

$$\gcd\left(f_0(a_j), f_1(a_j) + v(a_j)\right) \geq |h(a_j)| \geq 2^{-n-1}(n-1)!A^{-n}D.$$

On the other hand,

$$|f_i(a_j)| \leq 2A^nH_i \qquad \text{and} \qquad |v(a_j)| \leq 2A^nE$$

for each $j = 0, \ldots, n$ and $i = 0, 1$. Thus

$$2(|f_0(a_j)v(a_j)|)^{1/2} \leq (16H_0EA^{2n})^{1/2} \leq 2^{-n-1}(n-1)!A^{-n}D.$$

These inequalities show that Lemma 3.3 applies and Step 3 does indeed find the value $V_j = v(a_j)$. Thus Algorithm 3.5 works correctly. For any j, the set of all V_j in Step 3 can be computed in time polynomial in $n \log(H_0H_1\varepsilon^{-1})$, by Lemma 3.3. Finally, the number of possibilities for the vector $(V_0, \ldots, V_n)$ is polynomial in $(\log DH_1\varepsilon^{-1})^n$. □

We remark that for any $v \in \mathbb{Z}[x]$ it is easy to check whether $v \in \mathcal{V}$; thus Algorithm 3.5 is of Las Vegas type in this sense.

4. Future directions

Several natural questions are left open.

QUESTION 4.1. *Chart (some of) the white territory in Figure 2.3.*

There is a clear disparity between Theorem 2.3 where both inputs are perturbed and Theorem 3.6 where only one input is perturbed. In order to eliminate this distinction one has to study the underlying integer analog.

QUESTION 4.2. *Find an algorithm for the integer approximate gcd problem for perturbations of both inputs and apply it to the polynomial problem with respect to height.*

QUESTION 4.3. *Relax our constraints on solvability and/or obtain impossibility results.*

QUESTION 4.4. *Study variants of the approximate gcd problem for multivariate polynomials.*

5. Acknowledgements

We thank the referees for their useful comments. The first author's work was supported by the B-IT Foundation and the Land Nordrhein-Westfalen, and the third author's by ARC grant DP0556431. Thanks go to Daniel Loebenberger for help with the figures and to Mark Giesbrecht for useful discussions. An Extended Abstract of this paper appeared as von zur Gathen & Shparlinski (2008).

References

DARIO A. BINI & PAOLA BOITO (2007). Structured Matrix-Based Methods for Polynomial ϵ-gcd: Analysis and Comparisons. *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation ISSAC2007,* Waterloo, Ontario, Canada 9–16. URL `http://dx.doi.org/10.1145/1277548.1277551`.

IOANNIS Z. EMIRIS, ANDRÉ GALLIGO & HENRI LOMBARDI (1997). Certified approximate univariate GCDs. *Journal of Pure and Applied Algebra* **117&118**, 229–251. ISSN 0022-4049. URL `http://dx.doi.org/10.1016/S0022-4049(97)00013-3`.

Joachim von zur Gathen & Jürgen Gerhard (2003). *Modern Computer Algebra*. Cambridge University Press, Cambridge, UK, 2nd edition. ISBN 0-521-82646-2, 800 pages. URL http://cosec.bit.uni-bonn.de/science/mca.html. First edition 1999.

Joachim von zur Gathen & Igor E. Shparlinski (2008). Approximate Polynomial gcd: Small Degree and Small Height Perturbations. In *Proceedings of LATIN 2008,* Búzios, Rio de Janeiro, Brazil, Eduardo Sany Laber, Claudson Bornstein, Loana Tito Nogueira & Luerbio Faria, editors, number 4957 in Lecture Notes in Computer Science, 276–283. Springer-Verlag, Berlin, Heidelberg. ISSN 0302-9743. URL http://dx.doi.org/10.1007/978-3-540-78773-0.

Nick Howgrave-Graham (2001). Approximate integer common divisors. In *Cryptography and Lattices: International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001*, J. H. Silverman, editor, number 2146 in Lecture Notes in Computer Science, 51–66. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-42488-1. ISSN 0302-9743. URL http://www.springerlink.com/content/ak783wexe7ghp5db/.

N. Karcanias, S. Fatouros, M. Mitrouli & G. H. Halikias (2006). Approximate greatest common divisor of many polynomials, generalised resultants, and strength of approximation. *Computers & Mathematics with Applications* **51**, 1817–1830. ISSN 0898-1221. URL http://dx.doi.org/10.1016/j.camwa.2006.01.010.

N. K. Karmarkar & Y. N. Lakshman (1998). On approximate GCDs of univariate polynomials. *Journal of Symbolic Computation* **26**(6), 653–666. ISSN 0747-7171. URL http://dx.doi.org/10.1006/jsco.1998.0232.

Bingyu Li, Zhengfeng Yang & Lihong Zhi (2005). Fast Low Rank Approximation of a Sylvester Matrix by Structured Total Least Norm. *Journal of Japan Society for Symbolic and Algebraic Computation* **11**(3,4), 165–174. URL http://www.mmrc.iss.ac.cn/ lzhi/paper.html.

Victor Y. Pan (2001). Computation of Approximate Polynomial GCDs and an Extension. *Information and Computation* **167**, 71–85.

David Rupprecht (1999). An algorithm for computing certified approximate GCD of n univariate polynomials. *Journal of Pure and Applied Algebra* **139**(1-3), 255–284. URL http://dx.doi.org/10.1016/S0022-4049(99)00014-6.

Arnold Schönhage (1985). Quasi-GCD Computations. *Journal of Complexity* **1**, 118–137.