IRREDUCIBILITY AND GREATEST COMMON DIVISOR ALGORITHMS FOR SPARSE POLYNOMIALS

Michael Filaseta
Mathematics Department
University of South Carolina
Columbia, SC 29208
USA

Andrew Granville
Département de Mathématiques
Université de Montréal
Montréal QC H3C 3J7
Canada

Andrzej Schinzel
Institute of Mathematics
Polish Academy of Sciences
ul. Śniadeckich 8, 00-950
Warsaw, Poland

1 Introduction

Let $f(x) = \sum_{j=0}^{r} a_j x^{d_j} \in \mathbb{Z}[x]$ with each a_j nonzero and with $d_r > d_{r-1} > \cdots > d_1 > d_0 = 0$. For simplicity, we refer to the degree d_r of $f(x)$ as n. Observe that $r+1$ is the number of terms of $f(x)$. For convenience, we suppose both $n > 2$ and $r > 0$. The height H, as usual, denotes the maximum of the absolute values of the a_j.

The lattice base reduction algorithm of A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovasz [7] gives a factoring algorithm for $f(x)$ that runs in time that depends polynomially on $\log H$ and n. This clearly serves also as an irreducibility test for $f(x)$. One problem we address in this paper is the somewhat different issue of describing an irreducibility algorithm for sparse polynomials, that is where r is small compared to n. We view the input as being the list of $r+1$ coefficients a_j together with the list of $r+1$ exponents d_j. With this in mind, the input is of size $O\big(r(\log H + \log n)\big)$. We give an algorithm for this problem that runs in time that is polynomial in $\log n$ (but note that the dependence on r and $\log H$ in our arguments is not polynomial).

For $f(x) \in \mathbb{Q}[x]$, we define $\tilde{f} = x^n f(1/x)$. We say that $f(x)$ is *reciprocal* if $f(x) = \pm\tilde{f}(x)$. Otherwise, we say that $f(x)$ is nonreciprocal. We note that $f(x)$ is reciprocal if and only if the

2000 Mathematics Subject Classification: 11Y16, 12Y05, 68W30, 11C08, 11R09

The first author was supported by the National Science Foundation and the National Security Agency and the second author by the Natural Sciences and Engineering Research Council of Canada.

condition $f(\alpha) = 0$ for $\alpha \in \mathbb{C}$ implies that $\alpha \neq 0$ and $f(1/\alpha) = 0$. Our methods require the additional assumption that $f(x)$ is nonreciprocal. We establish the following.

Theorem A. *There is a constant $c_1 = c_1(r, H)$ such that an algorithm exists for determining whether a given nonreciprocal polynomial $f(x) \in \mathbb{Z}[x]$ as above is irreducible and that runs in time $O\big(c_1 \log n\,(\log\log n)^2 \log\log\log n\big)$.*

The result relies heavily on some recent work by E. Bombieri and U. Zannier described by the latter in an appendix of [11]. Alternatively, we can make use of [1], work by these same authors and D. Masser, which describes a new simplified approach to the previous work. The other main ingredients are the third author's application of the work of Bombieri and Zannier, given originally in [10], and an improvement on the the first and third authors' joint work in [4].

The constant c_1 can be made explicit. We note though that c_1 depends on some effectively computable constants that are not explicitly given in the appendix of [11] or in [1]. We therefore do not address this issue further here.

The algorithm will give, with the same running time, some information on the factorization of $f(x)$ in the case that $f(x)$ is reducible. Specifically, we have the following:

(i) If $f(x)$ has a cyclotomic factor, then the algorithm will detect this and output an $m \in \mathbb{Z}^+$ such that the cyclotomic polynomial $\Phi_m(x)$ divides $f(x)$.

(ii) If $f(x)$ does not have a cyclotomic factor but has a non-constant reciprocal factor, then the algorithm will produce such a factor. In fact, the algorithm will produce a reciprocal factor of $f(x)$ of maximal degree.

(iii) Otherwise, if $f(x)$ is reducible, then the algorithm outputs a complete factorization of $f(x)$ as a product of irreducible polynomials over $\mathbb{Q}$.

The algorithm for Theorem A will follow along the lines given above. First, we will check if $f(x)$ has a cyclotomic factor. If it does, the algorithm will produce m as in (i) and stop. If it does not, then the algorithm will check if $f(x)$ has a non-cyclotomic non-constant reciprocal factor. If it does, then the algorithm will produce such a factor as in (ii) and stop. If it does not, then the algorithm will output a complete factorization of $f(x)$ as indicated in (iii).

Our approach to (i) will allow us to obtain additional information about the complete set of cyclotomic factors of $f(x)$. In particular, we are able to describe, in the same running time given for the algorithm in Theorem A, the factor of $f(x)$ which has largest degree and only cyclotomic divisors. Details are given in the next section.

Our approach can be modified to show that if $f(x) \in \mathbb{Z}[x]$ is nonreciprocal and reducible, then $f(x)$ has a non-trivial factor in $\mathbb{Z}[x]$ containing $O(c_2)$ terms where $c_2 = c_2(r, H)$. We note that the results of [9] imply that if $f(x)$ also does not have a reciprocal factor, then every factor of $f(x)$ in $\mathbb{Z}[x]$ contains $O(c_2)$ terms.

In the case that $f(x) \in \mathbb{Z}[x]$ is reciprocal, one can modify our approach to obtain some information on the factorization of $f(x)$. Define the nonreciprocal part of $f(x)$ to be the polynomial $f(x)$ removed of its irreducible reciprocal factors in $\mathbb{Z}[x]$ with positive leading coefficients. Then in the case that $f(x)$ is reciprocal, one can still determine in time $O(c_1(\log n\,(\log\log n)^2 \log\log\log n)$ whether the nonreciprocal part of $f(x)$ is irreducible. Furthermore, in this same time, one can determine whether $f(x)$ has a cyclotomic factor and, if so, an integer m for which $\Phi_m(x)$ divides $f(x)$.

In addition, we address the problem of computing the greatest common divisor of two sparse polynomials. For nonzero $f(x)$ and $g(x)$ in $\mathbb{Z}[x]$, we use the notation $\gcd_{\mathbb{Z}}(f(x), g(x))$ to denote the polynomial in $\mathbb{Z}[x]$ of largest degree and largest positive leading coefficient that divides $f(x)$ and $g(x)$ in $\mathbb{Z}[x]$. Later in the paper, we will also make use of an analogous definition for $\gcd_{\mathbb{Z}}(f, g)$ where f and g are in $\mathbb{Z}[x_1, \dots, x_r]$. In this case, we interpret the leading coefficient as the coefficient of the expression $x_1^{e_1} x_2^{e_2} \dots x_r^{e_r}$ with e_1 maximal, then e_2 maximal given e_1, and so on. Our main result for the greatest common divisor of two sparse polynomials is the following.

Theorem B. *There is an algorithm which takes as input two polynomials $f(x)$ and $g(x)$ in $\mathbb{Z}[x]$, each of degree $\le n$ and height $\le H$ and having $\le r+1$ nonzero terms, with at least one of $f(x)$ and $g(x)$ free of cyclotomic factors, and outputs the value of $\gcd_{\mathbb{Z}}(f(x), g(x))$ and runs in time $O\big(c_3 \log n\big)$ for some constant $c_3 = c_3(r, H)$.*

Our approach will imply that if $f(x), g(x) \in \mathbb{Z}[x]$ are as above with $f(x)$ or $g(x)$ not divisible by a cyclotomic polynomial, then $\gcd_{\mathbb{Z}}(f(x), g(x))$ has $O(c_4)$ terms where $c_4 = c_4(r, H)$. The same conclusion does not hold if one omits the assumption that either $f(x)$ or $g(x)$ is not divisible by a cyclotomic polynomial. The following example, demonstrating this, was originally noted in the related work of the third author [12]. Let a and b be relatively prime positive integers. Then

$$\gcd\big(x^{ab} - 1, (x^a - 1)(x^b - 1)\big) = \frac{(x^a - 1)(x^b - 1)}{x - 1}.$$

In connection with Theorem B, we note that D. A. Plaisted [8] has shown that computing $\gcd_{\mathbb{Z}}(f(x), g(x))$ for general sparse polynomials $f(x)$ and $g(x)$ in $\mathbb{Z}[x]$ is at least as hard as any problem in NP. On the other hand, his proof relies heavily on considering polynomials $f(x)$ and $g(x)$ that have cyclotomic factors. By contrast, our proof of Theorem B will rest heavily on the fact that one of $f(x)$ or $g(x)$ does not have any cyclotomic factors.

Our proof of Theorem A will rely on Theorem B. In fact, Theorem B is where we make use of the work of Bombieri and Zannier already cited. It is possible to prove Theorem A in a slightly more direct way, for example by making use of Theorem 80 in [11] instead of Theorem B and Theorem 1 below. This does not avoid the use of the work of Bombieri and Zannier since Theorem 80 of [11] is based on this work. We have chosen the presentation here, however, because it clarifies that parts of the algorithm in Theorem A can rest on ideas that have been around for over forty years. In addition, we want the added information given by (i), (ii) and (iii) above as well as Theorem B itself.

To aid in our discussions, we have used letters for labelling theorems that establish the existence of an algorithm and will refer to the algorithms using the corresponding format. As examples, Algorithm A will refer to the algorithm given by Theorem A, and Algorithm B will refer to the algorithm given by Theorem B. Also, we make use of the notation $O_{r,H}\big(w(n)\big)$ to denote a function with absolute value bounded by $w(n)$ times a function of r and H. Thus, the running time for Algorithm A and Algorithm B can be expressed as $O_{r,H}\big(\log n\,(\log\log n)^2 \log\log\log n\big)$ and $O_{r,H}(\log n)$, respectively.

2 The Proof of Theorem A

We begin with the following result which improves on the main result in [4].

Theorem C. *There is an algorithm that has the following property: given $f(x) = \sum_{j=0}^{r} a_j x^{d_j} \in \mathbb{Z}[x]$ of degree $n > 1$ and with $r + 1 > 1$ terms, the algorithm determines whether $f(x)$ has a cyclotomic factor in running time $O_{r,H}\big(\log n\,(\log\log n)^2 \log\log\log n\big)$, where H denotes the height of $f(x)$. Furthermore, with the same running time, if $f(x)$ is divisible by a cyclotomic polynomial, then the algorithm outputs a positive integer m for which $\Phi_m(x)$ divides $f(x)$.*

Proof. We begin as in the proof of Theorem 2 of [4] and initially give an argument for the existence of an algorithm as in the theorem with running time $O_{r,H}\big((\log n)^2\big)$. We then explain how the algorithm can be sped up to produce the running time given in the statement of the theorem.

We describe and make use of Theorem 5 from [2]. For k a positive integer, define $\gamma(k) = 2 + \sum_{p|k}(p-2)$. Following [2], we call a vanishing sum S minimal if no proper subsum of S vanishes. We will be interested in sums $S = \sum_{j=1}^{t} a_j\omega_j$ where t is a positive integer, each a_j is a nonzero rational number and each ω_j is a root of unity. We refer to the reduced exponent of such an S as the least positive integer k for which $(\omega_i/\omega_1)^k = 1$ for all $i \in \{1, 2, \ldots, t\}$. Theorem 5 of [2] asserts then that if $S = \sum_{j=1}^{t} a_j\omega_j$ is a minimal vanishing sum, then $t \geq \gamma(k)$ where k is the reduced exponent of S. Also, note that Theorem 5 of [2] implies that the reduced exponent k of a minimal vanishing sum is necessarily squarefree.

To explain our algorithm, suppose first that $f(x)$ has a cyclotomic factor $\Phi_m(x)$, and that we can write $f(x) = \sum_{i=1}^{s} f_i(x)$ where each $f_i(x)$ is a nonzero polynomial divisible by $\Phi_m(x)$, no two $f_i(x)$ have terms involving x to the same power, and s is maximal. Observe that each $f_i(x)$ necessarily has at least two terms. Setting $\zeta_m = e^{2\pi i/m}$, we see that each $f_i(\zeta_m)$ is a minimal vanishing sum. For each $i \in \{1, 2, \ldots, s\}$, we write $f_i(x) = x^{b_i} g_i(x^{e_i})$ where $g_i(x) \in \mathbb{Z}[x]$, b_i and e_i are nonnegative integers chosen so that $g_i(0) \neq 0$ and the greatest common divisor of the exponents appearing in $g_i(x)$ is 1. Then $g_i(\zeta_m^{e_i})$ is a minimal vanishing sum with reduced exponent $m_i = m/\gcd(m, e_i)$. Necessarily, we have $g_i(\zeta_{m_i}) = 0$ and m_i is squarefree. Also, if t_i denotes the number of nonzero terms of $g_i(x)$, we have

$$t_i \geq \gamma\big(m_i\big) = 2 + \sum_{p|m_i}(p-2),$$

which implies each prime divisor of m_i is $\leq t_i$. Define

$$M_i = \{\ell \in \mathbb{Z}^+ : \Phi_\ell(x) \mid g_i(x),\ \ell \text{ is squarefree, and } \gamma(\ell) \leq t_i\}.$$

In particular, $m_i \in M_i$. In other words,

$$\frac{m}{\gcd(m, e_i)} \in M_i \quad \text{for all } i \in \{1, 2, \ldots, s\}. \tag{1}$$

We have not explained how we can write $f(x) = \sum_{i=1}^{s} f_i(x)$ as above. In particular, even if we know m exists with $\Phi_m(x)$ dividing $f(x)$, we do not know what m is. We circumvent this issue by considering every possible partition of the set $\{0, 1, \ldots, r\}$ as a disjoint union of sets $J_1, J_2, \ldots, J_s$ with each set J_i containing at least two elements. For each partition, we consider the polynomials

$$f_i(x) = \sum_{j \in J_i} a_j x^{d_j} = x^{b_i} g_i(x^{e_i}), \quad 1 \leq i \leq s,$$

where as before b_i and e_i are nonnegative integers chosen so that $g_i(0) \neq 0$ and the greatest common divisor of the exponents appearing in $g_i(x)$ is 1. Defining t_i and M_i as above, depending on the partition of $\{0, 1, \dots, r\}$, we see then that if $f(x)$ is divisible by some $\Phi_m(x)$, then there is a partition for which (1) holds. On the other hand, if (1) holds for some positive integer m and some partition of $\{0, 1, \dots, r\}$ as above, then we have $f_i(\zeta_m) = 0$ for each $i \in \{1, 2, \dots, s\}$, which implies $f(\zeta_m) = 0$ and hence $\Phi_m(x) \mid f(x)$. Thus, (1) holding for some m and some partition of $\{0, 1, \dots, r\}$ as above is a necessary and sufficient condition for $f(x)$ to be divisible by a cyclotomic polynomial.

With the above in mind, we describe the algorithm for determining whether $f(x)$ has a cyclotomic factor, give further justification that the algorithm works and give a proof that its running time is as claimed. The algorithm is as follows. We go through every partition of the set $\{0, 1, \dots, r\}$ into disjoint non-empty sets $J_1, J_2, \dots, J_s$ with each set J_i containing at least two elements. Observe that there are $O_r(1)$ such partitions. For each such partition and each $i \in \{1, 2, \dots, s\}$, we set $u = u(i)$ to be the element of J_i for which d_u is minimal. In terms of our definition of $f_i(x)$ and $g_i(x)$, this means $b_i = d_u$ and e_i is the greatest common divisor of the degrees of the terms of the polynomial $f_i(x)/x^{d_u}$. We compute e_i by taking the greatest common divisor of the numbers $d_v - d_u$ where $v \in J_i$. In terms of the complexity of the algorithm, given J_i, determining d_u can be done in $O_r(\log n)$ bit operations and computing e_i takes at most $O_r\big((\log n)^2\big)$ bit operations (cf. the discussion of Euclid's algorithm in [3, p. 79]). We can in fact obtain a running time of $O_r\big(\log n\,(\log\log n)^2 \log\log\log n\big)$ using a recursive gcd computation for large integers [3, p. 428] leading to the running time stated in Theorem C, but for the moment we use the $O_r\big((\log n)^2\big)$ estimate. The number of these computations that are needed as we vary over the partitions of $\{0, 1, \dots, r\}$ and vary over the sets J_i making up the partitions is $O_r(1)$. The computations have therefore thus far taken at most $O_r\big((\log n)^2\big)$ bit operations.

Next, for each partition $J_1, J_2, \dots, J_s$ of $\{0, 1, \dots, r\}$ as above, we compute the sets M_i as follows. Observe that t_i is the number of elements of J_i and is necessarily $\leq r + 1$. Thus, we can construct a list of the ℓ that are squarefree positive integers and such that $\gamma(\ell) \leq t_i$ in time $O_r(1)$. For each such ℓ, we want to check if $\Phi_\ell(x)$ divides $g_i(x)$. An algorithm that works well here and in more generality as well is given as Algorithm A in [4]. For our purposes, we can simply take each term $a_v x^{(d_v - d_u)/e_i}$ in $g_i(x)$, where $v \in J_i$, and replace it with $a_v x^{d'_v}$ where $d'_v \in \{0, 1, \dots, \ell - 1\}$ and

$$d'_v \equiv \frac{d_v - d_u}{e_i} \pmod{\ell}.$$

If we call the resulting polynomial $h_i(x)$, then $g_i(x)$ is divisible by $\Phi_\ell(x)$ if and only if $h_i(x)$ is divisible by $\Phi_\ell(x)$. Observe that the degree of $h_i(x)$ is $\leq \ell \leq (r + 1)^r$. Also, the height of $h_i(x)$ is $\leq (r + 1)H$. Hence, one can check directly if $h_i(x)$ is divisible by $\Phi_\ell(x)$ in time $O_{r,H}(1)$. The construction of each $h_i(x)$ takes time no more than $O_{r,H}\big((\log n)(\log\log n)^2\big)$, for $\varepsilon > 0$ arbitrary, where the main contribution of the time required comes from the division of $d_v - d_u$ by e_i above. Hence, the total time spent on constructing the various M_i as we vary over the partitions $J_1, J_2, \dots, J_s$ of $\{0, 1, \dots, r\}$ and $i \in \{1, 2, \dots, s\}$ is $O_{r,H}\big((\log n)(\log\log n)^2\big)$.

For the algorithm, we consider each partition $J_1, J_2, \dots, J_s$ of $\{0, 1, \dots, r\}$ as above one at a time. We construct the numbers e_i and the sets M_i as indicated. Next, we want to determine for a fixed partition whether (1) holds for some positive integer m. In other words, we want to know whether there is an m and $m_i \in M_i$ for which

$$(2) \qquad m = m_i \gcd(m, e_i) \quad \text{for } i \in \{1, 2, \dots, s\}.$$

For a positive integer k, we use the notation $\nu_p(k)$ to denote the positive integer u such that $p^u \| k$. Then (2) holds if and only if each of the following is true:

- If $p | m_1 \dots m_s$, then $\nu_p(m) \le \nu_p(m_i e_i)$ for all i with equality whenever p divides m_i.
- If $p \nmid m_1 \dots m_s$, then $\nu_p(m) \le \nu_p(e_0)$, where $e_0 = \gcd(e_1, \dots, e_s)$.

Defining

$$D = \prod_{\substack{p^t \| e_0 \\ p \nmid m_1 \cdots m_s}} p^t = e_0 \Big/ \Bigg(\prod_{\substack{p^t \| e_0 \\ p | m_1 \cdots m_s}} p^t \Bigg) \quad \text{and} \quad m_0 = \gcd(m_1 e_1, \dots, m_s e_s)/D,$$

then we see that a solution to (2) exists if and only if there exist m_i in M_i such that for every prime p dividing some m_i, the exact power of p dividing m_0 is the same as the exact power of p dividing $m_i e_i$. Furthermore, the set of m satisfying (2) in this case is precisely the set of $m = m_0 d$, where $d | D$. Observe that m_0 is the unique m satisfying (2) (if such m exist) with the property that every prime divisor of m is a divisor of $m_1 m_2 \cdots m_s$. Furthermore, every prime divisor of $m_1 m_2 \cdots m_s$ is a divisor of m_0. We are interested in knowing whether there exist m and m_i satisfying (2), so we simply restrict our attention to determining whether there exist m_i in M_i such that

$$(3) \qquad m_0 = m_i \gcd(m_0, e_i) \quad \text{for } i \in \{1, 2, \dots, s\}.$$

Recall that the numbers e_i and all elements of M_i have been computed (for each $i = 1, 2, \dots, s$). Also, as the partitions vary, the number of different e_i and m_i in M_i that arise is $O_r(1)$. We go through all these possibilities and compute $\mathcal{P}$, the set of primes dividing $m_1 m_2 \cdots m_s$. There are $O_r(1)$ such primes and it takes $O_r(1)$ time to compute them. We compute e_0, D and m_0 as defined above and check whether (3) holds. Note that the second formula for D involves removing the prime divisors from e_0 that are in $\mathcal{P}$, which is a fixed set of primes of size $O_r(1)$. Thus, both e_0 and D can be computed in time $O_r\big((\log n)^2\big)$. We also compute m_0 and check (3) with the same bound on the running time. If an m_0 is obtained for which (3) holds, then we output that $f(x)$ has a cyclotomic factor, indicate that the choice of $m = m_0$ is such that $\Phi_m(x)$ divides $f(x)$ and end the algorithm. If no m_0 is obtained for which (3) holds, then we output that $f(x)$ does not have a cyclotomic factor. As there are $O_r(1)$ different m_0 each of size $O_r(n)$, the running time estimate is not affected by going through the various m_0 and outputting the result. Hence, the proof of the theorem, but with running time only $O_{r,H}\big((\log n)^2\big)$, has been explained.

We improve the running time as follows. For the algorithm above, we made use of a few different greatest common divisor computations. These were done to construct e_i for $i \in \{1, 2, \dots, s\}$, to calculate $e_0 = \gcd(e_1, \dots, e_s)$ and $m_0 = \gcd(m_1 e_1, \dots, m_s e_s)/D$, and to determine the value of the right-hand side of (3). As noted earlier, we can apply known algorithms for gcd computations [3, p. 428] that would allow us to reduce the running time to that required by the theorem. However, it is also worth noting that these gcd computations can be circumvented and the required running time obtained in a different manner. We explain this approach now.

Let $J_1, J_2, \dots, J_s$ be a partition of $\{0, 1, \dots, r\}$ as in the argument above. Write $e_i = e_i' e_i''$ where every prime divisor of e_i' is $\le r+1$ and every prime divisor of e_i'' is $> r+1$. Recall that $u = u(i) \in J_i$ is chosen so that d_u is minimal. One can compute e_i' without computing e_i from the formula

$$e_i' = \prod_{p \le t_i} p^{\min_{v \in J_i}\{\nu_p(d_v - d_u)\}}.$$

In other words, for each $p \le t_i$, we can calculate the minimum of $\nu_p(d_v - d_u)$ as v runs through the elements of J_i and then form the product above to get e_i'. As we shall see momentarily, the numbers e_i' can be calculated in time $O_r(\log n\,(\log\log n)^2 \log\log\log n)$.

We note now that

$$g_i\bigl(x^{e_i''}\bigr) = \sum_{v \in J_i} a_v x^{(d_v - d_u)/e_i'},$$

so we can compute $g_i\bigl(x^{e_i''}\bigr)$ without computing $g_i(x)$, e_i or e_i''. Define

$$M_i' = \{\ell \in \mathbb{Z}^+ : \Phi_\ell(x) \mid g_i\bigl(x^{e_i''}\bigr),\ \ell \text{ is squarefree, and } \gamma(\ell) \le t_i\}.$$

The set M_i' can be computed in the same manner that we computed M_i but with $g_i(x)$ replaced by $g_i\bigl(x^{e_i''}\bigr)$. Thus, computing M_i', given the polynomials $g_i\bigl(x^{e_i''}\bigr)$, takes time $O_{r,H}\bigl((\log n)(\log\log n)^2\bigr)$. Recall that the prime divisors of e_i'' are all $> r+1 \ge t_i$. We deduce that the numbers ℓ in the definition of M_i and M_i' are relatively prime to e_i''. It follows that $M_i = M_i'$. Thus, the above analysis allows us to compute M_i without explicitly computing the numbers e_i and with running time $O_{r,H}\bigl(\log n\,(\log\log n)^2 \log\log\log n\bigr)$.

Next, we address how to determine whether (3) holds. Recall that $\mathcal{P}$ is the set of prime divisors of $m_1 m_2 \cdots m_s$, and note that these primes are $\le r+1$. The prime divisors of m_0 are precisely the primes in $\mathcal{P}$. We deduce that (3) holds if and only if

$$\nu_p(m_0) = \nu_p(m_i) + \min\{\nu_p(m_0), \nu_p(e_i)\} \tag{4}$$

for each $i \in \{1, 2, \dots, s\}$ and for each $p \in \mathcal{P}$. For each prime $p \in \mathcal{P}$, we compute the values of $\nu_p(e_i)$, for $i \in \{1, 2, \dots, s\}$, by using that $\nu_p(e_i) = \nu_p(e_i')$. Next, we compute

$$\nu_p(m_0) = \min_{1 \le i \le s}\{\nu_p(m_i) + \nu_p(e_i)\}.$$

Then we check if (4) holds. Observe that each $\nu_p(m_i)$ is either 0 or 1, so $\nu_p(m_i)$ can be computed by a simple division. We want also a method to compute $\nu_p(e_i) = \nu_p(e_i')$, for $i \in \{1, 2, \dots, s\}$. We further need to explain the computation of $\nu_p(d_v - d_u)$ to obtain e_i' above. For U a positive integer and p a prime $\le r+1$, the value of $\nu_p(U)$ can be computed as follows. We compute the values of p^{2^j} successively for $j \ge 0$ by squaring until we arrive at a positive integer t for which $p^{2^t} > U$. Observe that $t = O(\log\log U)$. We set $k_0 = 0$. For $j \in \{1, 2, \dots, t\}$, we successively check if $p^{2^{t-j}} | U$ and, if so, set $k_j = k_{j-1} + 2^{t-j}$ and replace U with $U/p^{2^{t-j}}$. If $p^{2^{t-j}} \nmid U$, then we set $k_j = k_{j-1}$. Then $k_t = \nu_p(U)$. Using this procedure, we can compute $\nu_p(U)$ in time $O_r(\log U\,(\log\log U)^2 \log\log\log U)$. The theorem follows. $\square$

Although it does not affect our main results, it is of some value to note that the running time of the algorithm can be shown to be $O_r\bigl(\log n\,(\log\log n)^2 \log\log\log n + \log H\bigr)$. Indeed, the coefficients of $f(x)$ only take part in the algorithm when we form the polynomials $h_i(x)$ and when we check their divisibility by $\Phi_\ell(x)$. Forming the polynomials involves $O_r(1)$ additions of these coefficients and checking the divisibility of an $h_i(x)$ by $\Phi_\ell(x)$ takes time $O_r(\log(H+1))$. Note that these divisions do not depend on n since the degrees and the coefficients of the polynomials are $O_r(1)$ and $O_r(H)$, respectively.

As it may be of interest in other contexts, we explain briefly how we can get a bit more out of the algorithm. More precisely, we explain how to obtain the largest monic factor $g(x)$ of $f(x)$ with each irreducible factor of $g(x)$ cyclotomic and in time $O_{r,H}\big(\log n\,(\log\log n)^2\log\log\log n\big)$. We begin with determining the product of the distinct cyclotomic divisors of $f(x)$. We note, however, that the representations of $g(x)$ and the product of the distinct cyclotomic divisors of $f(x)$ as polynomials cannot be the obvious ones as it is not difficult to show that for $a \ge 2$, the cyclotomic factors of $x^{(a-1)^2}+x^a-x-1$ are distinct and their product contains exactly $2a-2$ terms. In other words, explicitly writing out $g(x)$, for example, can take time considerably more than any power of $\log n$.

For given positive integers u, v, define the set $C(u,v) = \{ud : \ d|v\}$. In the algorithm above, we determined values m_0 and D such that $\Phi_m(x)$ divides $f(x)$ whenever $m \in C(m_0, D)$. Let S be the set of all such pairs $\{m_0, D\}$ that can arise as a solution to (2) in Algorithm C. We proved that $\Phi_m(x)$ divides $f(x)$ if and only if m is in the set

$$C_S = \bigcup_{\{m_0,D\}\in S} C(m_0, D).$$

We want to determine

$$\Phi_S(x) = \prod_{m\in C_S} \Phi_m(x).$$

The obvious way to do this is by determining each $C(m_0, D)$ explicitly, but that would involve factoring D which, for complexity issues, should be avoided. However, we can get around determining $C(m_0, D)$ explicitly by taking advantage of the fact that

$$\prod_{m\in C(u,v)} \Phi_m(x) = \Phi_u(x^v)$$

as follows.

We make a few observations about the sets $C(u,v)$:

- One has $C(U,V) \subseteq C(u,v)$ if and only if UV divides uv, u divides U and, as a consequence, V divides v.

- Given positive integers u, v, u', v' with $\gcd(u,v) = \gcd(u',v') = 1$, define $U = \mathrm{lcm}(u,u')$, and let $V = \gcd(v,v')$. Note that $\gcd(U,V) = 1$. Then

$$C(u,v) \cap C(u',v') = \begin{cases} C(U,V) & \text{if } UV \text{ divides } \gcd(uv, u'v') \\ \emptyset & \text{otherwise.} \end{cases}$$

- There is a natural ordering on the pairs $\{u,v\}$ where $u, v \in \mathbb{N}$, taking $\{U,V\} < \{u,v\}$ if $UV < uv$, or if $UV = uv$ and $V < v$. We see that if $C(U,V) \subset C(u,v)$ then $\{U,V\} < \{u,v\}$.

Now $|S| = O_r(1)$. Given S we create a new set T. We start with $T_0 = S$, and then recursively construct

$$T_{k+1} = \{\{U, V\} : \ C(U, V) = C(u, v) \cap C(u', v') \text{ for some } \{u, v\}, \ \{u', v'\} \in T_k\} \cup T_k.$$

One can show that $T_{k+1} = T_k$ for some $k = O_r(1)$. When $T_{k+1} = T_k$, we set $T = T_k$. Note that $|T| = O_r(1)$ and $\gcd(u, v) = 1$ for all $\{u, v\} \in T$. For each $\{u, v\} \in T$, beginning with uv minimal and v minimal given uv, we define the polynomials

$$\Phi_{\{u,v\}}(x) = \Phi_u(x^v) \Big/ \prod_{\substack{\{U,V\} \in T \\ \{U,V\} < \{u,v\} \\ C(U,V) \subset C(u,v)}} \Phi_{\{U,V\}}(x) \ \in \mathbb{Z}[x].$$

We do not compute these polynomials explicitly but can give their values as the quotient above where $\{u, v\}$ and each $\{U, V\}$ in the product are given explicitly. Then we have

$$\Phi_S(x) = \prod_{\{u,v\} \in T} \Phi_{\{u,v\}}(x).$$

Obtaining this description of $\Phi_S(x)$ takes $O_r\big(\log n \, (\log\log n)^2 \log\log\log n\big)$ bit operations.

The polynomial $\Phi_S(x)$ is the product of all the distinct cyclotomic factors of $f(x)$. To deal with cyclotomic factors to higher multiplicities, we make use of the following lemma due to G. Hajós [5] (also, see [11, p. 187]).

Lemma 1. *If $(x - \alpha)^k$ divides $f(x)$, then $k \le r$.*

Recall that S was defined as the set of $\{m_0, D\}$ that gave rise to solutions of (2) corresponding to cyclotomic factors of $f(x)$. We construct similar sets S_j corresponding to cyclotomic factors of $f^{(j)}(x)$ for every $j \in \{0, 1, \dots, r-1\}$. Observe that the coefficients of $f^{(j)}(x)$ are bounded by $n^j H$, the degree of $f^{(j)}(x)$ is $n - j$ (assuming as we can that $n \ge r$) and the number of terms in $f^{(j)}(x)$ is $\le r + 1$. Recalling that the running time of Algorithm C is $O_r\big(\log n \, (\log\log n)^2 \log\log\log n + \log H\big)$, it is not difficult to see that the running time for computing the various sets S_j is $O_{r,H}\big(\log n \, (\log\log n)^2 \log\log\log n\big)$. The exact multiplicity of a cyclotomic factor of $f(x)$ is k provided it divides $f^{(j)}(x)$ for $0 \le j \le k-1$ and not $f^{(k)}(x)$. Lemma 1 further implies that if a cyclotomic polynomial divides $f^{(j)}(x)$ for every $j \in \{0, 1, \dots, r-1\}$, then the multiplicity of the factor is r (i.e., there is no need to check if the cyclotomic factor divides $f^{(r)}(x)$). However, we need to be able to determine the common cyclotomic factors determined by various sets S_j. To do this, we set $S_0^* = S_0$, and then construct recursively

$$S_{k+1}^* = \{\{U, V\} : \ C(U, V) = C(u, v) \cap C(u', v') \text{ for some } \{u, v\} \in S_k^*, \ \{u', v'\} \in S_{k+1}\}$$

for each $k \in \{1, 2, \dots, r-1\}$. One can then proceed by determining T_k^* from S_k^* as we constructed T from S above, and then compute $\Phi_{S_k^*}(x)$, the product of the distinct cyclotomic polynomials dividing $f(x)$ with multiplicity at least $k+1$. The product of the polynomials $\Phi_{S_k^*}(x)$ for $k \in \{0, 1, \dots, r-1\}$ is therefore the largest degree factor of $f(x)$ that is a product of cyclotomic polynomials. The total running time is $O_{r,H}\big(\log n \, (\log\log n)^2 \log\log\log n\big)$ for describing this factor of $f(x)$.

We are now ready to return to our description of Algorithm A. Algorithm A begins by taking the input polynomial $f(x)$ and applying Algorithm C. If $f(x)$ has a cyclotomic factor, we obtain m as in (i). As $f(x)$ is not reciprocal, $f(x)$ cannot be a constant multiple of a cyclotomic polynomial. Hence, $f(x)$ is reducible and (i) holds.

This part of the algorithm does not actually depend on $f(x)$ being nonreciprocal. The proof of Algorithm C shows in fact that if $f(x)$ has a cyclotomic factor, then one can determine m as in (i) with every prime divisor of m being $\leq r+1$. Thus, it would not be difficult to factor m and compute $\phi(m)$ in the running time required for Theorem A. Once $\phi(m)$ is computed, then one can determine if $f(x)$ is a constant multiple of the cyclotomic polynomial $\Phi_m(x)$ by comparing $\phi(m)$ with n.

We suppose now that $f(x)$ does not have a cyclotomic factor. The next step in Algorithm A is to determine whether $f(x)$ has a reciprocal factor. We shall do this by making use of Theorem B, which we establish in the next section.

We compute

$$\tilde{f}(x) = x^n f(1/x) = \sum_{j=0}^{r} a_j x^{n-d_j}.$$

Since $f(x)$ does not have a cyclotomic factor, we can apply Algorithm B to compute $h(x) = \gcd_{\mathbb{Z}}(f(x), \tilde{f}(x))$. Observe that $h(x)$ is reciprocal and each reciprocal factor of $f(x)$ divides $h(x)$. As $f(x)$ is not reciprocal, we must have $\deg h < \deg f$. If $h(x)$ is not constant, then $f(x)$ is reducible, $h(x)$ is a non-constant reciprocal factor of $f(x)$ and (ii) holds as $h(x)$ is a reciprocal polynomial of largest possible degree dividing $f(x)$. Otherwise, $f(x)$ does not have a non-constant reciprocal factor. Theorem B implies that this part of Algorithm A has running time $O_{r,H}\big(\log n\big)$.

We are now left with considering the case that $f(x)$ does not have any non-constant reciprocal factor. The basic idea here is to make use of the third author's work in [9] (see also Theorem 74 in [11]). For a polynomial $F\big(x_1, \ldots, x_r, x_1^{-1}, \ldots, x_r^{-1}\big)$, in the variables $x_1, \ldots, x_r$ and their reciprocals $x_1^{-1}, \ldots, x_r^{-1}$, we define

$$J\,F = x_1^{u_1} \cdots x_r^{u_r} F\big(x_1, \ldots, x_r, x_1^{-1}, \ldots, x_r^{-1}\big),$$

where each u_j is an integer chosen as small as possible so that $J\,F$ is a polynomial in $x_1, \ldots, x_r$. In the way of examples, if

$$F = x^2 + 4x^{-1}y + y^3 \quad \text{and} \quad G = 2xyw - x^2z^{-3}w - 12w,$$

then

$$J\,F = x^3 + 4y + xy^3 \quad \text{and} \quad J\,G = 2xyz^3 - x^2 - 12z^3.$$

In particular, note that although w is a variable in G, the polynomial $J\,G$ does not involve w. We call a multi-variable polynomial $F(x_1, \ldots, x_r) \in \mathbb{Q}[x_1, \ldots, x_r]$ *reciprocal* if

$$J\,F\big(x_1^{-1}, \ldots, x_r^{-1}\big) = \pm F(x_1, \ldots, x_r).$$

For example, $x_1x_2 - x_1 - x_2 + 1$ and $x_1x_2 - x_3x_4$ are reciprocal. Note that this is consistent with our definition of a reciprocal polynomial $f(x) \in \mathbb{Z}[x]$.

To motivate the next result and begin our approach, we set

$$F(x_1, \ldots, x_r) = a_r x_r + \cdots + a_1 x_1 + a_0 \in \mathbb{Z}[x_1, \ldots, x_r].$$

The plan is to associate the factorization of $f(x) = F(x^{d_1}, x^{d_2}, \dots, x^{d_r})$ with the factorization of a multi-variable polynomial of the form

$$\tilde{F}\big(y_1^{m_{11}} \cdots y_t^{m_{1t}}, \dots, y_1^{m_{r1}} \cdots y_t^{m_{rt}}\big),$$

where the number of variables t is $\le r$ and $m_{ij} \in \mathbb{Z}$ for $1 \le i \le r$ and $1 \le j \le t$. The above multi-variable polynomial can be expressed as

$$y_1^{u_1} \cdots y_t^{u_t} F(y_1^{m_{11}} \cdots y_t^{m_{1t}}, \dots, y_1^{m_{r1}} \cdots y_t^{m_{rt}}),$$

where

$$u_j = -\min\{m_{1j}, m_{2j}, \dots, m_{rj}\} \quad \text{for } 1 \le j \le t. \tag{5}$$

To make the connection with the factorization of $f(x)$, we want the matrix $M = (m_{ij})$ to be such that

$$\begin{pmatrix} d_1 \\ \vdots \\ d_r \end{pmatrix} = M \begin{pmatrix} v_1 \\ \vdots \\ v_t \end{pmatrix} \tag{6}$$

for some integers $v_1, v_2, \dots, v_t$. In this way, the substitution $y_j = x^{v_j}$ for $1 \le j \le t$ takes any factorization

$$y_1^{u_1} \cdots y_t^{u_t} F(y_1^{m_{11}} \cdots y_t^{m_{1t}}, \dots, y_1^{m_{r1}} \cdots y_t^{m_{rt}}) = F_1(y_1, \dots, y_t) \cdots F_s(y_1, \dots, y_t) \tag{7}$$

in $\mathbb{Z}[y_1, \dots, y_t]$ into the form

$$x^{u_1 v_1 + \cdots + u_t v_t} F(x^{d_1}, x^{d_2}, \dots, x^{d_r}) = F_1(x^{v_1}, \dots, x^{v_t}) \cdots F_s(x^{v_1}, \dots, x^{v_t}). \tag{8}$$

We restrict our attention to factorizations in (7) where the $F_i(y_1, \dots, y_t)$ are non-constant. We will be interested in the case that s is maximal; in other words, we will want the right-hand side of (7) to be a complete factorization of the left-hand side of (7) into irreducibles over $\mathbb{Q}$. For achieving the results in this paper, we want some algorithm for obtaining such a complete factorization of multi-variable polynomials; among the various sources for this, we note that A. K. Lenstra's work in [6] provides such an algorithm. For the moment, though, we need not take s maximal.

Since $f(x) = F(x^{d_1}, x^{d_2}, \dots, x^{d_r})$, the above describes a factorization of $f(x)$, except that we need to take some caution as some v_j may be negative so the expressions $F_i(x^{v_1}, \dots, x^{v_t})$ may not be polynomials in x. For $1 \le i \le s$, define w_i as the integer satisfying

$$\tilde{F}_i(x^{v_1}, \dots, x^{v_t}) = x^{w_i} F_i(x^{v_1}, \dots, x^{v_t}). \tag{9}$$

We obtain from (8) that

$$x^{u_1 v_1 + \cdots + u_t v_t + w_1 + \cdots + w_s} f(x) = \prod_{i=1}^{s} x^{w_i} F_i(x^{v_1}, \dots, x^{v_t}).$$

The definition of w_i implies that this product is over polynomials in $\mathbb{Z}[x]$ that are not divisible by x. The conditions $a_0 \ne 0$ and $d_0 = 0$ imposed on $f(x)$ in the introduction imply that $f(x)$ is not divisible by x. Hence, the exponent of x appearing on the left must be 0, and we obtain the factorization

$$f(x) = \prod_{i=1}^{s} x^{w_i} F_i(x^{v_1}, \dots, x^{v_t}) = \prod_{i=1}^{s} J\, F_i(x^{v_1}, \dots, x^{v_t}). \tag{10}$$

The factorization given in (10) is crucial to our algorithm. As we are interested in the case that $f(x)$ has no non-constant reciprocal factor, we restrict our attention to this case. From (10), we see that the polynomials $x^{w_i} F_i(x^{v_1}, \dots, x^{v_t})$ cannot have a non-constant reciprocal factor. There are, however, still two possibilities that we need to consider for each $i \in \{1, 2, \dots, s\}$:

(i′) $F_i(y_1, \dots, y_t)$ is reciprocal.

(ii′) $J\, F_i(x^{v_1}, \dots, x^{v_t}) \in \mathbb{Z}$.

Although we will not need to know a connection between (i′) and (ii′), we show here that if (i′) holds for some i, then (ii′) does as well. We consider then the possibility that

$$J\, F_i\big(y_1^{-1}, \dots, y_t^{-1}\big) = \pm F_i(y_1, \dots, y_t). \tag{11}$$

In other words, suppose that

$$y_1^{e_1} \cdots y_t^{e_t} F_i\big(y_1^{-1}, \dots, y_t^{-1}\big) = \pm F_i(y_1, \dots, y_t), \tag{12}$$

where $e_j = e_j(i)$ is the degree of $F_i(y_1, \dots, y_t)$ as a polynomial in y_j. Substituting $y_j = x^{v_j}$ into (12), we obtain

$$x^{w_i + e_1 v_1 + \cdots + e_t v_t} F_i\big(x^{-v_1}, \dots, x^{-v_t}\big) = \pm x^{w_i} F_i\big(x^{v_1}, \dots, x^{v_t}\big). \tag{13}$$

By the definition of w_i, the polynomial on the right does not vanish at 0. Assume (ii′) does not hold. Let α be a zero of this polynomial. Then substituting $x = 1/\alpha$ into (13) shows that $1/\alpha$ is also a zero. On the other hand, we have already demonstrated in (10) that the right-hand side of (13) is a factor of $f(x)$. This contradicts that $f(x)$ has no non-constant reciprocal factor. Hence, (ii′) holds.

We make use of a special case of a result due to the third author in [9]. In particular, the more general result implies that the above idea can in fact always be used to factor $f(x)$ if $f(x)$ has two nonreciprocal irreducible factors. In other words, there exist a matrix M and v_j satisfying (6) and a factorization of the form (7) that leads to a non-trivial factorization of $f(x)$, if it exists, through the substitution $y_j = x^{v_j}$. We are interested in the case that $f(x)$ has no non-constant reciprocal factor. In this case, we can obtain a complete factorization of $f(x)$ into irreducibles.

Theorem 1. *Fix*

$$F = F(x_1, \dots, x_r) = a_r x_r + \cdots + a_1 x_1 + a_0,$$

where the a_j are nonzero integers. There exists a finite computable set of matrices S with integer entries, depending only on F, with the following property: Suppose the vector $\overrightarrow{d} = \langle d_1, d_2, \dots, d_r \rangle$ is in $\mathbb{Z}^r$ with $d_r > d_{r-1} > \cdots > d_1 > 0$ and such that $f(x) = F(x^{d_1}, x^{d_2}, \dots, x^{d_r})$ has no non-constant reciprocal factor. Then there is an $r \times t$ matrix $M = (m_{ij}) \in S$ of rank $t \le r$ and a vector $\overrightarrow{v} = \langle v_1, v_2, \dots, v_t \rangle$ in $\mathbb{Z}^t$ such that (6) holds and the factorization given by (7) in $\mathbb{Z}[y_1, \dots, y_t]$ of a polynomial in t variables $y_1, y_2, \dots, y_t$ as a product of s irreducible polynomials over $\mathbb{Q}$ implies the factorization of $f(x)$ given by (10) as a product of polynomials in $\mathbb{Z}[x]$ each of which is either irreducible over $\mathbb{Q}$ or a constant.

We are ready now to apply the above to assist us in Algorithm A. As suggested by the statement of Theorem 1, we take the coefficients a_j of $f(x)$ and consider the multi-variable polynomial $F = F(x_1, \dots, x_r)$. We compute the set S. Since F is a linear polynomial with $r+1$ terms and height H, the time required to compute S is $O_{r,H}(1)$. Since $f(x) = F(x^{d_1}, \dots, x^{d_r})$ has no non-constant reciprocal factors, there is a matrix $M = (m_{ij}) \in S$ of rank $t \le r$ and a vector $\overrightarrow{v}$ in $\mathbb{Z}^t$ as in Theorem 1. We go through each of the $O_{r,H}(1)$ matrices M in S and solve for the vectors $\overrightarrow{v} = \langle v_1, v_2, \dots, v_t \rangle$ in $\mathbb{Z}^t$ satisfying $\overrightarrow{d} = M\overrightarrow{v}$, where t is the number of columns in M and we interpret $\overrightarrow{d}$ and $\overrightarrow{v}$ as column vectors. From the definition of S, we have that the rank of M is t and $t \le r$. Hence, there can be at most one such vector $\overrightarrow{v}$ for each $M \in S$. However, for each $\overrightarrow{d}$, there may be many $M \in S$ and $\overrightarrow{v}$ for which $\overrightarrow{d} = M\overrightarrow{v}$, and we will consider all of them.

We make use of the following simple result in this section and the next.

Theorem D. *There is an algorithm with the following property. Given an $r \times t$ integral matrix $M = (m_{ij})$ of rank $t \le r$ and $\max\{|m_{ij}|\} = O_{r,H}(1)$ and given an integral vector $\overrightarrow{d} = \langle d_1, \dots, d_r \rangle$ with $\max\{|d_j|\} = O_{r,H}(n)$, the algorithm determines whether there is an integral vector $\overrightarrow{v} = \langle v_1, \dots, v_t \rangle$ for which (6) holds, and if such a $\overrightarrow{v}$ exists, the algorithm outputs the solution vector $\overrightarrow{v}$. Furthermore, $\max\{|v_j|\} = O_{r,H}(n)$ and the algorithm runs in time $O_{r,H}(\log n)$.*

Proof. There are a variety of ways we can determine if $\overrightarrow{d} = M\overrightarrow{v}$ has a solution and to determine the solution if there is one within the required time $O_{r,H}(\log n)$. We use Gaussian elimination. Performing elementary row operations on M and multiplying by entries from the matrix as one proceeds to use only integer arithmetic allows us to rewrite M in the form of an $r \times t$ matrix $M' = (m'_{ij})$ with each $m'_{ij} \in \mathbb{Z}$ and the first t rows of M' forming a $t \times t$ diagonal matrix with nonzero integers along the diagonal. These computations only depend on the entries of M and, hence, take time $O_{r,H}(1)$. We perform the analogous row operations and integer multiplications on the vector $\overrightarrow{d} = \langle d_1, d_2, \dots, d_r \rangle$ to solve $\overrightarrow{d} = M\overrightarrow{v}$ for $\overrightarrow{v}$. As the entries of M are integers that are $O_{r,H}(1)$ and each d_j is an integer that is $O_{r,H}(n)$, these operations take time $O_{r,H}(\log n)$. We are thus left with an equation of the form $\overrightarrow{d'} = M'\overrightarrow{v}$ where the entries of M' are integers that are $O_{r,H}(1)$ and the components of $\overrightarrow{d'} = \langle d'_1, d'_2, \dots, d'_r \rangle$ are integers that are $O_{r,H}(n)$.

For each $j \in \{1, 2, \dots, t\}$, we check if $d'_j \equiv 0 \pmod{m'_{jj}}$. If for some $j \in \{1, 2, \dots, t\}$ we have $d'_j \not\equiv 0 \pmod{m'_{jj}}$, then a solution to the original equation $\overrightarrow{d} = M\overrightarrow{v}$, if it exists, must be such that $v_j \notin \mathbb{Z}$. In this case, an integral vector $\overrightarrow{v}$ does not exist. Now, suppose instead that $d'_j \equiv 0 \pmod{m'_{jj}}$ for every $j \in \{1, 2, \dots, t\}$. Then we divide d'_j by m'_{jj} to determine the vector $\overrightarrow{v}$. This vector may or may not be a solution to the equation $\overrightarrow{d} = M\overrightarrow{v}$. We check whether it is by a direct computation. If it is not a solution to the equation $\overrightarrow{d} = M\overrightarrow{v}$, then there are no solutions to the equation. Otherwise, $\overrightarrow{v}$ is an integral vector satisfying $\overrightarrow{d} = M\overrightarrow{v}$. Checking whether $d'_j \equiv 0 \pmod{m'_{jj}}$ for $1 \le j \le t$, solving for $\overrightarrow{v}$ if it holds, and checking whether $\overrightarrow{d} = M\overrightarrow{v}$ all takes time $O_{r,H}(\log n)$. We also have $O_{r,H}(n)$ as a bound for the absolute value of the components v_j of $\overrightarrow{v}$. We output $\overrightarrow{v}$ if it exists which takes time $O_{r,H}(\log n)$. Combining the running times above, the theorem follows. □

Algorithm D is performed for each of the $O_{r,H}(1)$ matrices M in S. The running time for each application of Theorem D is $O_{r,H}(\log n)$, so the total running time spent applying Algorithm D for the various $O_{r,H}(1)$ matrices in S is $O_{r,H}(\log n)$. This leads to $O_{r,H}(1)$ factorizations of the form given in (7) into irreducibles, each having a potentially different value for s. For each of these, we compute the values of $F_i\big(x^{v_1}, \dots, x^{v_t}\big)$ and determine w_i as in (9). We produce then $O_{r,H}(1)$ factorizations of $f(x)$ as in (10). As we obtain these factorizations, we keep track of the number of non-constant polynomials $x^{w_i} F_i\big(x^{v_1}, \dots, x^{v_t}\big)$ appearing in (10). We choose a factorization for which this number is maximal. Recalling that (10) follows from $\overrightarrow{d} = M\overrightarrow{v}$ and (7), we deduce from Theorem 1 that the factorization of $f(x)$ we have chosen provides a factorization of $f(x)$ with each $x^{w_i} F_i\big(x^{v_1}, \dots, x^{v_t}\big)$ either irreducible or constant. Recalling that the polynomials $F_i(y_1, \dots, y_t)$ in (7) are independent of n and that the components of $\overrightarrow{v}$ are bounded in absolute value by $O_{r,H}(n)$, we see that producing the factorization of $f(x)$ into irreducibles and constants as in (10) takes time $O_{r,H}(\log n)$. For a factorization of $f(x)$ into irreducibles over $\mathbb{Q}$, we multiply together the constants appearing on the right of (10) and one of the irreducible polynomials $J\,F_i\big(x^{v_1}, \dots, x^{v_t}\big)$. This does not affect the bound given for the running time of Algorithm A.

Thus, we have demonstrated an algorithm for Theorem A as stated in the introduction and justified that the algorithm satisfies the statement of Theorem A as well as (i), (ii) and (iii). Combining the above running time estimates, we deduce that the algorithm also has the stated running time bound given in Theorem A.

3 The Proof of Theorem B

As mentioned in the introduction, our proof of Theorem B relies heavily on the recent work of Bombieri and Zannier outlined by Zannier in an appendix in [11]. In particular, as a direct consequence of their more general work, we have

Theorem 2. *Let*

$$F(x_1, \dots, x_k), G(x_1, \dots, x_k) \in \mathbb{Q}[x_1, \dots, x_k]$$

be coprime polynomials. There exists an effectively computable number $B(F, G)$ with the following property. If $\overrightarrow{u} = \langle u_1, \dots, u_k \rangle \in \mathbb{Z}^k$, $\xi \neq 0$ is algebraic and

$$F(\xi^{u_1}, \dots, \xi^{u_k}) = G(\xi^{u_1}, \dots, \xi^{u_k}) = 0,$$

then either ξ is a root of unity or there exists a nonzero vector $\overrightarrow{v} \in \mathbb{Z}^k$ having components bounded in absolute value by $B(F, G)$ and orthogonal to $\overrightarrow{u}$.

It is important for our algorithm that the quantities $B(F, G)$ are effectively computable. We note that the fact $B(F, G)$ is effectively computable is not explicitly stated in the appendix of [11], but U. Zannier (private communication) has pointed out that the approach given there does imply that this is the case. The more recent paper [1] notes explicitly that $B(F, G)$ can be calculated.

Our description of Algorithm B has similarities to the third author's application of Theorem 2 in [10] and [11]. In particular, we make use of the following lemma which is Corollary 6 in Appendix E of [11]. A proof is given there.

Lemma 2. *Let ℓ be a positive integer and $\overrightarrow{v} \in \mathbb{Z}^\ell$ with $\overrightarrow{v}$ nonzero. The lattice of vectors $\overrightarrow{u} \in \mathbb{Z}^\ell$ orthogonal to $\overrightarrow{v}$ has a basis $\overrightarrow{v_1}', \overrightarrow{v_2}', \dots, \overrightarrow{v_{\ell-1}}'$ such that the maximum absolute value of a component of any vector $\overrightarrow{v_j}'$ is bounded by $\ell/2$ times the maximum absolute value of a component of $\overrightarrow{v}$.*

For our algorithm, we can suppose that $f(x)$ does not have a cyclotomic factor and do so. We consider only the case that $f(0)g(0) \neq 0$ as computing $\gcd_{\mathbb{Z}}(f(x), g(x))$ can easily be reduced to this case by initially removing an appropriate power of x from each of $f(x)$ and $g(x)$ (that is, by subtracting the least degree of a term from each exponent). This would need to be followed up by possibly multiplying by a power of x after our gcd computation.

We furthermore only consider the case that the content of $f(x)$, that is the greatest common divisor of its coefficients, and the content of $g(x)$ are 1. Otherwise, we simply divide by the contents before proceeding and then multiply the final result by the greatest common divisor of the two contents.

We express our two polynomials in the form

$$f(x) = \sum_{j=0}^{k} a_j x^{d_j} \quad \text{and} \quad g(x) = \sum_{j=0}^{k} b_j x^{d_j},$$

where above we have possibly extended the lists of exponents and coefficients describing $f(x)$ and $g(x)$ so that the exponent lists are identical and the coefficient lists are allowed to include coefficients which are 0. Also, we take $d_k > d_{k-1} > \cdots > d_1 > 0$. Thus, $d_0 = 0$, $a_0 b_0 \neq 0$ and $k \leq 2r$. The time required to modify $f(x)$ and $g(x)$ so that they are not divisible by x and have content 1 and to adjust the exponent and coefficient lists as above is $O_{r,H}(\log n)$.

Before continuing with the algorithm, we motivate it with some discussion. Let $w(x)$ denote $\gcd_{\mathbb{Z}}(f(x), g(x))$. We will apply Theorem 2 to construct two finite sequences of polynomials in several variables F_u and G_u with integer coefficients and a corresponding finite sequence of vectors $\overrightarrow{d}^{(u)}$ that will enable us to determine a polynomial in $\mathbb{Z}[x]$ that has the common zeros, to the correct multiplicity, of $f(x)$ and $g(x)$. This then will allow us to compute $w(x)$.

Let ξ be a zero of $w(x)$, if it exists. Observe that $\xi \neq 0$, and since ξ is a zero of $f(x)$ which has no cyclotomic factors, we have ξ is not a root of unity. Since ξ is a common zero of $f(x)$ and $g(x)$, we have

$$\sum_{j=0}^{k} a_j \xi^{d_j} = \sum_{j=0}^{k} b_j \xi^{d_j} = 0.$$

We recursively construct F_u, G_u and $\overrightarrow{d}^{(u)}$, for $0 \leq u \leq s$, where s is to be determined, beginning with

$$F_0 = F_0(x_1, \ldots, x_k) = \sum_{j=0}^{k} a_j x_j \quad \text{and} \quad G_0 = G_0(x_1, \ldots, x_k) = \sum_{j=0}^{k} b_j x_j, \tag{14}$$

and $\overrightarrow{d}^{(0)} = \langle d_1, d_2, \ldots, d_k \rangle$. As u increases, the number of variables defining F_u and G_u will decrease. The value of s then will be $\leq k$. Observe that

$$F_0(x^{d_1}, \ldots, x^{d_k}) = f(x) \quad \text{and} \quad G_0(x^{d_1}, \ldots, x^{d_k}) = g(x).$$

We deduce that F_0 and G_0, being linear, are coprime in $\mathbb{Q}[x_1, \ldots, x_k]$ and that

$$F_0(\xi^{d_1}, \ldots, \xi^{d_k}) = G_0(\xi^{d_1}, \ldots, \xi^{d_k}) = 0. \tag{15}$$

Now, suppose for some $u \geq 0$ that nonzero polynomials F_u and G_u in $\mathbb{Z}[x_1, \dots, x_{k_u}]$ and a vector $\overrightarrow{d}^{(u)} = \langle d_1^{(u)}, \dots, d_{k_u}^{(u)} \rangle \in \mathbb{Z}^{k_u}$ have been determined, where $k_u < k_{u-1} < \cdots < k_0 = k$. Furthermore, suppose that F_u and G_u are coprime in $\mathbb{Q}[x_1, \dots, x_{k_u}]$ and that we have at least one zero ξ of $w(x)$ such that

$$F_u\big(\xi^{d_1^{(u)}}, \dots, \xi^{d_{k_u}^{(u)}}\big) = G_u\big(\xi^{d_1^{(u)}}, \dots, \xi^{d_{k_u}^{(u)}}\big) = 0. \tag{16}$$

In particular, $\xi \neq 0$ and ξ is not a root of unity. Note that the $d_j^{(u)}$ may be negative. We will require

$$J\,F_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}}) \mid f(x) \quad \text{and} \quad J\,G_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}}) \mid g(x). \tag{17}$$

Observe that $J\,F_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}})$ and $f(x)$ are in $\mathbb{Z}[x]$. We take (17) to mean that there is a polynomial $h(x) \in \mathbb{Z}[x]$ such that

$$f(x) = h(x) \cdot J\,F_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}})$$

with an analogous equation holding for $g(x)$ and $J\,G_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}})$. In particular, we want $J\,F_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}})$ and $J\,G_u(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}})$ to be nonzero. Note that these conditions which are being imposed on F_u and G_u are satisfied for $u = 0$ provided $w(x)$ is not constant. For $0 \leq u < s$, we describe next how to recursively construct F_{u+1} and G_{u+1} having analogous properties. The specifics of the algorithm and its running time will be discussed later.

There is a computable bound $B(F_u, G_u)$ as described in Theorem 2. We deduce that there is a nonzero vector $\overrightarrow{v} = \langle v_1, v_2, \dots, v_{k_u} \rangle \in \mathbb{Z}^{k_u}$ such that each $|v_i| \leq B(F_u, G_u)$ and $\overrightarrow{v}$ is orthogonal to $\overrightarrow{d}^{(u)}$. From Lemma 2, there is a $k_u \times (k_u - 1)$ matrix $\mathcal{M}$ with each entry of $\mathcal{M}$ having absolute value $\leq k_u B(F_u, G_u)/2$ and such that $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ for some $\overrightarrow{v}^{(u)} \in \mathbb{Z}^{k_u - 1}$, where we view the vectors as column vectors. We define integers m_{ij} (written also $m_{i,j}$) and $v_j^{(u)}$, depending on u, by the conditions

$$\mathcal{M} = \begin{pmatrix} m_{11} & \cdots & m_{1,k_u-1} \\ \vdots & \ddots & \vdots \\ m_{k_u 1} & \cdots & m_{k_u,k_u-1} \end{pmatrix} \quad \text{and} \quad \overrightarrow{v}^{(u)} = \langle v_1^{(u)}, \dots, v_{k_u-1}^{(u)} \rangle.$$

The relations

$$x_i = y_1^{m_{i1}} \cdots y_{k_u-1}^{m_{i,k_u-1}} \quad \text{for } 1 \leq i \leq k_u$$

transform the polynomials $F_u(x_1, \dots, x_{k_u})$ and $G_u(x_1, \dots, x_{k_u})$ into polynomials in some, possibly all, of the variables $y_1, \dots, y_{k_u-1}$. These new polynomials we call $\mathcal{F}_u$ and $\mathcal{G}_u$, respectively. More precisely, we define

$$\mathcal{F}_u(y_1, \dots, y_{k_u-1}) = J\,F_u\big(y_1^{m_{11}} \cdots y_{k_u-1}^{m_{1,k_u-1}}, \dots, y_1^{m_{k_u 1}} \cdots y_{k_u-1}^{m_{k_u,k_u-1}}\big) \tag{18}$$

and

$$\mathcal{G}_u(y_1, \dots, y_{k_u-1}) = J\,G_u\big(y_1^{m_{11}} \cdots y_{k_u-1}^{m_{1,k_u-1}}, \dots, y_1^{m_{k_u 1}} \cdots y_{k_u-1}^{m_{k_u,k_u-1}}\big). \tag{19}$$

The polynomials $\mathcal{F}_u$ and $\mathcal{G}_u$ will depend on the matrix $\mathcal{M}$ so that there may be many choices for $\mathcal{F}_u$ and $\mathcal{G}_u$ for each F_u and G_u. We need only consider one such $\mathcal{F}_u$ and $\mathcal{G}_u$ and do so. Note that this still may require considering various $\mathcal{M}$ until we find one for which $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ is satisfied for some $\overrightarrow{v}^{(u)} \in \mathbb{Z}^{k_u-1}$. The equation $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ implies that for some integers $e_f(u)$ and $e_g(u)$ we have

$$\mathcal{F}_u\big(x^{v_1^{(u)}}, \dots, x^{v_{k_u-1}^{(u)}}\big) = x^{e_f(u)} F_u\big(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}}\big) \tag{20}$$

and

$$\mathcal{G}_u\big(x^{v_1^{(u)}}, \dots, x^{v_{k_u-1}^{(u)}}\big) = x^{e_g(u)} G_u\big(x^{d_1^{(u)}}, \dots, x^{d_{k_u}^{(u)}}\big). \tag{21}$$

In particular, $\mathcal{F}_u$ and $\mathcal{G}_u$ are nonzero. Also,

$$J\,\mathcal{F}_u\big(x^{v_1^{(u)}}, \dots, x^{v_{k_u-1}^{(u)}}\big) \mid f(x) \quad \text{and} \quad J\,\mathcal{G}_u\big(x^{v_1^{(u)}}, \dots, x^{v_{k_u-1}^{(u)}}\big) \mid g(x). \tag{22}$$

Furthermore, with ξ as in (16), we have

$$\mathcal{F}_u\big(\xi^{v_1^{(u)}}, \dots, \xi^{v_{k_u-1}^{(u)}}\big) = \mathcal{G}_u\big(\xi^{v_1^{(u)}}, \dots, \xi^{v_{k_u-1}^{(u)}}\big) = 0.$$

The idea is to suppress the variables, if they exist, which do not occur in $\mathcal{F}_u$ and $\mathcal{G}_u$ and the corresponding components of $\overrightarrow{v}^{(u)}$ to obtain the polynomials F_{u+1} and G_{u+1} and the vector $\overrightarrow{d}^{(u+1)}$ for our recursive construction. However, there is one other matter to consider. The polynomials $\mathcal{F}_u$ and $\mathcal{G}_u$ may not be coprime, and we require F_{u+1} and G_{u+1} to be coprime. Hence, we adjust this idea slightly.

Let

$$D_u = D_u(y_1, \dots, y_{k_u-1}) = \gcd_{\mathbb{Z}}(\mathcal{F}_u, \mathcal{G}_u) \in \mathbb{Z}[y_1, \dots, y_{k_u-1}]. \tag{23}$$

Recall that $f(0)g(0) \neq 0$. Hence, (20), (21) and (22) imply that $J\,D_u\big(x^{v_1^{(u)}}, \dots, x^{v_{k_u-1}^{(u)}}\big)$ divides $\gcd_{\mathbb{Z}}(f, g)$ in $\mathbb{Z}[x]$. We define

$$F_{u+1} = \frac{\mathcal{F}_u(y_1, \dots, y_{k_u-1})}{D_u(y_1, \dots, y_{k_u-1})} \quad \text{and} \quad G_{u+1} = \frac{\mathcal{G}_u(y_1, \dots, y_{k_u-1})}{D_u(y_1, \dots, y_{k_u-1})}, \tag{24}$$

and set $k_{u+1} \leq k_u - 1$ to be the total number of variables $y_1, \dots, y_{k_u-1}$ appearing in F_{u+1} and G_{u+1}. Note that F_{u+1} and G_{u+1} are coprime and that (17) holds with u replaced by $u+1$ and the appropriate change of variables.

We describe next how the recursive construction will end. Suppose we have just constructed F_u, G_u and $\overrightarrow{d}^{(u)}$ and proceed as above to the next step of constructing F_{u+1}, G_{u+1} and $\overrightarrow{d}^{(u+1)}$. At this point, D_{u-1} will have been defined but not D_u. We want to find $\mathcal{M}$ and a $\overrightarrow{v}^{(u)}$ such that $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ where $\mathcal{M}$ is a $k_u \times (k_u - 1)$ matrix with entries bounded in absolute value by $k_u B(F_u, G_u)/2$. So we compute $B(F_u, G_u)$ and the bound $k_u B(F_u, G_u)/2$ on the absolute values of the entries of $\mathcal{M}$. We consider such $\mathcal{M}$ and apply Algorithm D to see if there is an integral vector $\overrightarrow{v}^{(u)}$ for which $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$. Once such an $\mathcal{M}$ and $\overrightarrow{v}^{(u)}$ are found, we can proceed with the construction of F_{u+1} and G_{u+1} given above. On the other hand, it is possible that no such

$\mathcal{M}$ and $\overrightarrow{v}^{(u)}$ will be found. Given Theorem 2, this will be the case only if the supposition that (16) holds for some zero ξ of $w(x)$ is incorrect. In particular, (16) does not hold for some zero ξ of $w(x)$ if F_u and G_u are coprime polynomials in < 2 variables (i.e., $k_u \le 1$), but it is also possible that (16) does not hold for some u with F_u and G_u polynomials in ≥ 2 variables (i.e., $k_u \ge 2$). Given that $\mathcal{M}$ is a $k_u \times (k_u - 1)$ matrix, we consider it to be vacuously true that no $\mathcal{M}$ and $\overrightarrow{v}^{(u)}$ exist satisfying $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ in the case that $k_u \le 1$. If no such $\mathcal{M}$ and $\overrightarrow{v}^{(u)}$ exist, we consider the recursive construction of the polynomials F_u and G_u complete and set $s = u$. We will want the values of D_u for every $1 \le u \le s-1$, so we save these as we proceed.

The motivation discussed above can be summarized into a procedure to be used for Algorithm B as follows. Beginning with F_0 and G_0 as in (14) and $\overrightarrow{d}^{(0)} = \langle d_1, \dots, d_k \rangle$, we construct the multi-variable polynomials F_u and G_u and vectors $\overrightarrow{d}^{(u)} = \langle d_1^{(u)}, \dots, d_{k_u}^{(u)} \rangle \in \mathbb{Z}^{k_u}$ recursively. Given F_u, G_u and $\overrightarrow{d}^{(u)}$, we compute $B(F_u, G_u)$ and search for a $k_u \times (k_u - 1)$ matrix $\mathcal{M}$ with integer entries having absolute value $\le k_u B(F_u, G_u)/2$ for which $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ is solvable with $\overrightarrow{v}^{(u)} = \langle v_1^{(u)}, \dots, v_{k_u-1}^{(u)} \rangle \in \mathbb{Z}^{k_u-1}$. We check for solvability and determine the solution $\overrightarrow{v}^{(u)}$ if it exists by using Algorithm D. If no such $\mathcal{M}$ and $\overrightarrow{v}^{(u)}$ exist, then we set $s = u$ and stop our construction. Otherwise, once such an $\mathcal{M} = (m_{ij})$ and $\overrightarrow{v}^{(u)}$ are determined, we define F_{u+1} and G_{u+1} using (18), (19), (23) and (24). After using (24) to construct F_{u+1} and G_{u+1}, we determine the variables $y_1, \dots, y_{k_u-1}$ which occur in F_{u+1} and G_{u+1} and define $\overrightarrow{d}^{(u+1)}$ as the vector with corresponding components from $v_1^{(u)}, \dots, v_{k_u-1}^{(u)}$; in other words, if y_j is the ith variable occurring in F_{u+1} and G_{u+1}, then $v_j^{(u)}$ is the ith component of $\overrightarrow{d}^{(u+1)}$.

For the running time for this recursive construction, we use that $B(F_u, G_u)$ is $O_{r,H}(1)$ as u varies and, furthermore, the numbers $B(F_u, G_u)$ can be computed in time $O_{r,H}(1)$. In particular, this implies that for a fixed u, there are $O_{r,H}(1)$ choices for $\mathcal{M}$ and, hence, a total of $O_{r,H}(1)$ possible values for F_{u+1} and G_{u+1} independent of the value of $\overrightarrow{d}^{(u)}$. In other words, without even knowing the values of $d_1, \dots, d_k$, we can use Theorem 2 to deduce that there are at most $O_{r,H}(1)$ possibilities for F_1 and G_1. For each of these possibilities, another application of Theorem 2 implies that there are at most $O_{r,H}(1)$ possibilities for F_2 and G_2. And so on. As $s \le k \le 2r$, we deduce that the total number of matrices $\mathcal{M}$ that we need to consider during the recursive construction is bounded by $O_{r,H}(1)$. The recursive construction depends on n only when applying Theorem D to see if $\overrightarrow{d}^{(u)} = \mathcal{M}\overrightarrow{v}^{(u)}$ holds for some $\overrightarrow{v}^{(u)}$ and to determine $\overrightarrow{v}^{(u)}$ if it exists. For a fixed $\mathcal{M}$, Theorem D implies that these computations can be done in time $O_{r,H}(\log n)$. As the total number of $\mathcal{M}$ to consider is bounded by $O_{r,H}(1)$, we deduce that the recursive construction of the F_u, G_u and $\overrightarrow{d}^{(u)}$ takes time $O_{r,H}(\log n)$.

As we proceed in our recursive construction of the F_u and G_u, an important aspect of the construction is that the m_{ij} are bounded in absolute value by $O_{r,H}(1)$ and, hence, the coefficients and exponents appearing in F_u and G_u are bounded by $O_{r,H}(1)$. In other words, F_u and G_u can be written in time $O_{r,H}(1)$. Another important aspect of the construction is to note that as we are dividing by D_u to construct F_{u+1} and G_{u+1}, we obtain not simply that $J\,D_u\big(x^{v_1^{(u)}}, \dots, x^{v_{k_u-1}^{(u)}}\big)$ divides $\gcd_{\mathbb{Z}}(f,g)$ in $\mathbb{Z}[x]$ but also

$$\prod_{j=0}^{u} J\,D_j\big(x^{v_1^{(j)}}, \dots, x^{v_{k_j-1}^{(j)}}\big) \text{ divides } \gcd_{\mathbb{Z}}(f,g) \text{ in } \mathbb{Z}[x]. \tag{25}$$

This can be seen inductively by observing that

$$J\,\mathcal{F}_u\big(x^{v_1^{(u)}},\dots,x^{v_{k_u-1}^{(u)}}\big)=\frac{f(x)}{\displaystyle\prod_{j=0}^{u-1} J\,D_j\big(x^{v_1^{(j)}},\dots,x^{v_{k_j-1}^{(j)}}\big)} \tag{26}$$

and

$$J\,\mathcal{G}_u\big(x^{v_1^{(u)}},\dots,x^{v_{k_u-1}^{(u)}}\big)=\frac{g(x)}{\displaystyle\prod_{j=0}^{u-1} J\,D_j\big(x^{v_1^{(j)}},\dots,x^{v_{k_j-1}^{(j)}}\big)}. \tag{27}$$

Algorithm B ends by making use of the identity

$$\gcd_{\mathbb{Z}}\big(f(x),g(x)\big)=\prod_{u=0}^{s-1} J\,D_u\big(x^{v_1^{(u)}},\dots,x^{v_{k_u-1}^{(u)}}\big). \tag{28}$$

We justify (28). Recall that we have denoted the left side by $w(x)$. Observe that (25) implies that the expression on the right of (28) divides $w(x)$. By the definition of s, when we arrive at $u=s$ in our recursive construction, (16) fails to hold for every zero ξ of $w(x)$. Therefore, taking $u=s-1$ in (24), (26) and (27) implies that the right-hand side of (28) vanishes at all the zeros of $w(x)$ and to the same multiplicity. As noted earlier, we are considering the case that the contents of $f(x)$ and $g(x)$ are 1. We deduce that (28) holds. Observe that the computation of $\gcd_{\mathbb{Z}}\big(f(x),g(x)\big)$ by expanding the right side of (28) involves $O_{r,H}(1)$ additions of exponents of size $O_{r,H}(n)$. This computation can be done in time $O_{r,H}(\log n)$. Theorem B follows.

References

[1] E. Bombieri, D. Masser and U. Zannier, *Anomalous subvarieties - structure theorems and applications*, preprint.

[2] J. H. Conway and A. J. Jones, *Trigonometric Diophantine equations (On vanishing sums of roots of unity)*, Acta Arith. **30** (1976), 229–240.

[3] R. Crandall and C. Pomerance, Prime Numbers, A Computational Perspective, Springer-Verlag, New York, 2001.

[4] M. Filaseta and A. Schinzel, *On testing the divisibility of lacunary polynomials by cyclotomic polynomials*, Math. Comp. **73** (2004), 957–965.

[5] G. Hajós, Solution of Problem 41 (Hungarian), Mat. Lapok 4 (1953), 40–41.

[6] A. K. Lenstra, *Factoring multivariate polynomials over algebraic number fields*, SIAM J. Comput. **16** (1987), 591–598.

[7] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), 515–534.

[8] D. A. Plaisted, *Sparse complex polynomials and polynomial reducibility*, J. Comput. System Sci. **14** (1977), 210–221.

[9] A. Schinzel, *Reducibility of lacunary polynomials, I*, Acta Arith. **16** (1969/1970), 123–159.

[10] A. Schinzel, *Reducibility of lacunary polynomials, XII*, Acta Arith. **90** (1999), 273–289.

[11] A. Schinzel, Polynomials with Special Regard to Reducibility, Encyclopedia of Mathematics and its Applications, 77, Cambridge University Press, 2000.

[12] A. Schinzel, *On the greatest common divisor of two univariate polynomials, I*, A panorama of number theory or the view from Baker's garden (Zürich, 1999), Cambridge Univ. Press, Cambridge, 2002, 337–352.