D. Harvey, M. Hittmeir *Res. Number Theory (2022)8:94*
https://doi.org/10.1007/s40993-022-00387-w

Research in Number Theory

RESEARCH

A deterministic algorithm for finding r-power divisors

David Harvey[1*] and Markus Hittmeir[2]

*Correspondence: d.harvey@unsw.edu.au
[1] School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia
Full list of author information is available at the end of the article

The first author was supported by the Australian Research Council (grant FT160100219). SBA Research (SBA-K1) is a COMET Centre within the framework of COMET - Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna. The COMET Programme is managed by FFG

Abstract

Building on work of Boneh, Durfee and Howgrave-Graham, we present a deterministic algorithm that provably finds all integers p such that $p^r \mid N$ in time $O(N^{1/4r+\epsilon})$ for any $\epsilon > 0$. For example, the algorithm can be used to test squarefreeness of N in time $O(N^{1/8+\epsilon})$; previously, the best rigorous bound for this problem was $O(N^{1/6+\epsilon})$, achieved via the Pollard–Strassen method.

1 Introduction

1.1 Statement of main result

Let r be a positive integer. In this paper we study the problem of finding all r-power divisors of a given positive integer N, i.e., all positive integers p such that $p^r \mid N$. Throughout the paper we write $\lg x := \log_2 x$, and unless otherwise specified, the "running time" of an algorithm refers to the number of bit operations it performs, or more formally, the number of steps executed by a deterministic multitape Turing machine [12]. We always assume the use of fast (quasilinear time) algorithms for basic integer arithmetic, i.e., for multiplication, division and GCD (see for example [19] or [4]).

Our main result is the following theorem.

Theorem 1.1 *There is an explicit deterministic algorithm with the following properties. It takes as input an integer $N \geqslant 2$ and a positive integer $r \leqslant \lg N$. Its output is a list of all positive integers p such that $p^r \mid N$. Its running time is*

$$O\left(N^{1/4r} \cdot \frac{(\lg N)^{10+\epsilon}}{r^3}\right). \tag{1.1}$$

Note that whenever we write ϵ in a complexity bound, we mean that the bound holds for all $\epsilon > 0$, where the implied big-O constant may depend on ϵ.

The integers p referred to in Theorem 1.1 need not be prime. Of course, if p is a composite integer found by the algorithm, then the algorithm will incidentally determine the complete factorisation of p, as the prime divisors ℓ of p must also satisfy $\ell^r \mid N$.

The hypothesis $r \leqslant \lg N$ does not really limit the applicability of the theorem: if $r > \lg N$ then the problem is trivial, as the only possible r-power divisor is 1.

Theorem 1.1 is intended primarily as a theoretical result. For fixed r the complexity is $O(N^{1/4r+\epsilon})$, which is fully exponential in $\lg N$, so the algorithm cannot compete asymptotically with subexponential factoring algorithms such as the elliptic curve method (ECM) or the number field sieve (NFS). Furthermore, experiments confirm that for small r the algorithm is grossly impractical compared to general-purpose factoring routines implemented in modern computer algebra systems.

1.2 Previous work

At the core of our algorithm is a generalisation of Coppersmith's method [6] introduced by Boneh, Durfee and Howgrave-Graham [1]. We refer to the latter as the *BDHG algorithm*. Coppersmith's seminal work showed how to use lattice methods to quickly find all divisors of N in certain surprisingly large intervals. To completely factor N, one simply applies the method to a sequence of intervals that covers all possible divisors up to $N^{1/2}$. Each interval is searched in polynomial time, so the overall complexity is governed by the number of such intervals, which turns out to be $O(N^{1/4+\epsilon})$. The BDHG algorithm adapts Coppersmith's method to the case of r-power divisors. The relationship between our algorithm and the BDHG algorithm is discussed in Sect. 1.3 below.

We emphasise that, unlike factoring algorithms such as ECM or NFS, whose favourable running time analyses depend on heuristic assumptions, the complexity bound in Theorem 1.1 is *rigorously analysed* and fully *deterministic*. Under these restrictions, for $r \geqslant 2$ it is asymptotically superior to all previously known complexity bounds for the problem of finding r-power divisors.

Its closest competitors are the algorithms of Strassen [17] and Pollard [14]. These algorithms can be used to find all divisors of N less than a given bound B in time $O(B^{1/2+\epsilon})$. If $p^r \mid N$, say $N = p^r q$, then either $p \leqslant N^{1/(r+1)}$ or $q \leqslant N^{1/(r+1)}$, so the Pollard–Strassen method can be used to find p or q, and hence both, in time $O(N^{1/2(r+1)+\epsilon})$. For example, taking $r = 2$, these algorithms can find all *square* divisors of N in time $O(N^{1/6+\epsilon})$, whereas our algorithm finds all square divisors in time $O(N^{1/8+\epsilon})$.

There is one special case in which the Pollard–Strassen approach still wins. If one knows in advance that p is relatively small, say $p < N^c$ for some $c \in (0, 1/2r)$, then the Pollard–Strassen method has complexity $O(N^{c/2+\epsilon})$, which is better than the bound in Theorem 1.1. Our algorithm can also take advantage of the information that $p < N^c$, but unfortunately this yields only a constant-factor speedup.

Another point of difference is the space complexity. The space required by the algorithm in Theorem 1.1 is only polynomial in $\lg N$ (we will not give the details of this analysis), whereas for the Pollard–Strassen method the space complexity is the same as the time complexity, up to logarithmic factors.

In connection with the case $r = 2$, two other works are worth mentioning. Booker, Hiary and Keating [3] describe a subexponential time algorithm that can sometimes prove that a given integer N is squarefree, with little or no knowledge of its factorisation. This algorithm is not fully rigorous, as its analysis depends on (among other things) the Generalised Riemann Hypothesis. Peralta and Okamoto [13] present a speedup of the ECM method for integers of the form $N = p^2 q$. Again this result is not fully rigorous, because it depends on standard conjectures concerning the distribution of smooth numbers in short intervals, just as in Lenstra's original ECM algorithm.

The case $r = 1$ corresponds to the ordinary factoring problem, and in this case our algorithm is essentially equivalent to Coppersmith's method. As mentioned above, the complexity is $O(N^{1/4+\epsilon})$, which does not improve on known results; currently, the fastest known deterministic factoring method has complexity $O(N^{1/5+\epsilon})$ [9]. (It is interesting to ask whether the ideas behind [9] can be used to improve Theorem 1.1 when $r \geqslant 2$. Our inquiries in this direction have been so far unsuccessful.)

In fact, when $r = 1$, Theorem 1.1 gives the more precise complexity bound $O(N^{1/4}(\lg N)^{10+\epsilon})$. It is apparently well known that Coppersmith's method has complexity $O(N^{1/4}(\lg N)^C)$ for some constant $C > 0$, but to the best of our knowledge, this is the first time in the literature that a particular value of C has been specified. On the other hand, we have not tried particularly hard to optimise the value of C, and it is likely that it can be improved. (One possible improvement is outlined in Remark 3.6.)

1.3 Relationship to the BDHG algorithm

The authors of [1] were mainly interested in cryptographic applications, and this led them to focus on the case that $N = p^r q$ where p and q are roughly the same size. In this setting, they show that their algorithm is faster than ECM when $r \approx (\log p)^{1/2}$, and that it even runs in polynomial time when r is as large as $\log p$.

In this paper we take a different point of view: our goal is to determine the worst-case complexity, without any assumptions on the size of p, q or r.

To illustrate what difference this makes, consider again the case $r = 2$. This case is mentioned briefly in Section 6 of [1]. The authors point out that if $N = p^2 q$, where p and q are known to be about the same size, i.e., both p and q are within a constant factor of $N^{1/3}$, then the running time of their method is $O(N^{1/9+\epsilon})$, i.e., the number of search intervals is $O(N^{1/9+\epsilon})$. However, in our more general setup, this is *not* the worst case. Rather, the worst case running time is $O(N^{1/8+\epsilon})$, which occurs when searching for $p \sim N^{1/4}$ and $q \sim N^{1/2}$.

More generally, for $r \geqslant 1$ the worst case running time of $O(N^{1/4r+\epsilon})$ stated in Theorem 1.1 occurs when $p \sim N^{1/2r}$ and $q \sim N^{1/2}$. By contrast, in the "balanced" situation considered in [1], where $p, q \sim N^{1/(r+1)}$, one can show that the running time is only $O(N^{1/(r+1)^2+\epsilon})$ (see Remark 3.5, and take $\theta = r/(r+1)$).

Although the core of our algorithm is essentially the same as the BDHG algorithm, our more general perspective requires us to make a few changes to their presentation. For instance, we cannot take the lattice dimension to be $d \approx r^2$ (as is done in the main theorem of [1]), because this choice is suboptimal when r is small and fixed. Additional analysis is required to deal with potentially small values of p and q, and in general we must take more care than [1] in estimating certain quantities throughout the argument. For these reasons, we decided to give a self-contained presentation, not relying on the results in [1].

Remark 1.2 After this paper was accepted for publication, Dan Bernstein mentioned to us (personal communication) that the proof of Theorem 5.2 of [2] likely includes much of the argument needed to obtain our Theorem 1.1.

1.4 Root-finding

An important component of our algorithm, and of all algorithms pursuing Coppersmith's strategy, is a subroutine for finding all integer roots of a polynomial with integer coefficients. This problem has received extensive attention in the literature, but we were unable to locate a clear statement of a deterministic complexity bound suitable for our purposes. For completeness, in Appendix A we give a detailed proof of the following result. For a polynomial $f \in \mathbb{Z}[x]$, we write $\|f\|_\infty$ for the maximum of the absolute values of the coefficients of f.

Theorem 1.3 *Let $b \geqslant n \geqslant 1$ be integers. Given as input a polynomial $f \in \mathbb{Z}[x]$ of degree n such that $\|f\|_\infty \leqslant 2^b$, we may find all of the integer roots of f in time*

$$O(n^{2+\epsilon} b^{1+\epsilon}).$$

Note that this complexity bound is much stronger than what is needed for the application in this paper. However, it is still not quasilinear in the size of the input, which is $O(nb)$. For further discussion, see Remarks A.8 and A.10.

2 Searching one interval

In this section we recall the strategy of [1] for finding all integers p in a prescribed interval $P - H \leqslant p \leqslant P + H$ such that $p^r \mid N$, provided that H is not too large. We will prove the following theorem.

Theorem 2.1 *There is an explicit deterministic algorithm with the following properties. It takes as input positive integers N, r, m, d, P and H such that*

$$r \leqslant \lg N, \tag{2.1}$$

$$m \leqslant d/r, \tag{2.2}$$

$$H < P \leqslant N^{1/r}, \tag{2.3}$$

and

$$H^{(d-1)/2} < \frac{1}{d^{1/2}\, 2^{(d-1)/4}} \cdot \frac{(P-H)^{rm}}{N^{rm(m+1)/2d}}. \tag{2.4}$$

Its output is a list of all integers p in the interval $P - H \leqslant p \leqslant P + H$ such that $p^r \mid N$. Its running time is

$$O\big(d^{7+\epsilon} (\tfrac{1}{r} \lg N)^{2+\epsilon}\big).$$

A key tool needed in the proof of Theorem 2.1 is the LLL algorithm:

Lemma 2.2 *Let $d \geqslant 1$ and $B \geqslant 2$. Given as input linearly independent vectors $v_0, \ldots, v_{d-1} \in \mathbb{Z}^d$ such that $\|v_i\| \leqslant B$, in time*

$$O\big(d^{5+\epsilon} (\lg B)^{2+\epsilon}\big)$$

we may find a nonzero vector w in the lattice $L := \operatorname{span}_{\mathbb{Z}}(v_0, \ldots, v_{d-1})$ such that

$$\|w\| \leqslant 2^{(d-1)/4} (\det L)^{1/d}.$$

(Here $\|\cdot\|$ denotes the standard Euclidean norm on $\mathbb{R}^d$.)

Proof We take w to be the first vector in a reduced basis for L computed by the LLL algorithm [10, Prop. 1.26]. For the bound on $\|w\|$, see [10, Prop. 1.6]. (For more recent developments on lattice reduction, see for example [8, Ch. 17].) □

Let $\mathbb{Z}[x]_d$ denote the space of polynomials in $\mathbb{Z}[x]$ of degree less than d. The first step in the proof of Theorem 2.1 is the following proposition, which uses the LLL algorithm to construct a nonzero polynomial $h \in \mathbb{Z}[x]_d$ with relatively small coefficients in a carefully chosen lattice.

Proposition 2.3 *Let N, r, m, d, P and H be positive integers satisfying (2.1), (2.2) and (2.3). Define polynomials $f_0, \dots, f_{d-1} \in \mathbb{Z}[x]_d$ by*

$$f_i(x) := \begin{cases} N^{m-\lfloor i/r \rfloor}(P+x)^i, & 0 \leqslant i < rm, \\ (P+x)^i, & rm \leqslant i < d. \end{cases}$$

Then in time

$$O\big(d^{7+\epsilon}(\tfrac{1}{r}\lg N)^{2+\epsilon}\big) \tag{2.5}$$

we may find a nonzero polynomial

$$h(x) = h_0 + \cdots + h_{d-1}x^{d-1} \in \mathbb{Z}[x]_d$$

in the $\mathbb{Z}$-span of $f_0, \dots, f_{d-1}$ such that

$$|h_0| + |h_1|H + \cdots + |h_{d-1}|H^{d-1} < d^{1/2}\, 2^{(d-1)/4} H^{(d-1)/2} N^{rm(m+1)/2d}. \tag{2.6}$$

Proof Set $\tilde{f}_i(y) := f_i(Hy) \in \mathbb{Z}[y]_d$, and let $v_i \in \mathbb{Z}^d$ be the vector whose j-th entry (for $j = 0, \dots, d-1$) is the coefficient of y^j in $\tilde{f}_i(y)$. We will apply Lemma 2.2 to the vectors $v_0, \dots, v_{d-1}$.

Let

$$B := d^{1/2}\, 2^d N^{d/r+1}.$$

We claim that $\|v_i\| \leqslant B$ for all i. First consider the case $0 \leqslant i < rm$. For any $j = 0, \dots, i$, the coefficient of y^j in $\tilde{f}_i(y) = N^{m-\lfloor i/r \rfloor}(P+Hy)^i$ is equal to

$$N^{m-\lfloor i/r \rfloor}\tbinom{i}{j}P^{i-j}H^j \leqslant N^{m-i/r+1}2^iP^i \leqslant 2^iN^{m+1} \leqslant 2^dN^{d/r+1},$$

where we have used the hypotheses (2.3) and (2.2). For the case $rm \leqslant i < d$, a similar argument shows that every coefficient of $\tilde{f}_i(y) = (P+Hy)^i$ is bounded above by $2^dN^{d/r}$. Therefore every v_i has coordinates bounded by $2^dN^{d/r+1}$, and we conclude that $\|v_i\| \leqslant B$ for all i.

Next we calculate the determinant of the lattice $L := \operatorname{span}_{\mathbb{Z}}(v_0, \dots, v_{d-1})$, or equivalently, the determinant of the $d \times d$ integer matrix whose rows are given by the v_i. Since $\deg \tilde{f}_i(y) = i$, this is a lower triangular matrix whose diagonal entries are given by the leading coefficients of the $\tilde{f}_i(y)$, namely

$$\begin{cases} N^{m-\lfloor i/r \rfloor}H^i, & 0 \leqslant i < rm, \\ H^i, & rm \leqslant i < d. \end{cases}$$

The determinant is the product of these leading coefficients, i.e.,

$$\begin{aligned} \det L &= H^{1+2+\cdots+(d-1)}\underbrace{(N^m \cdots N^m)}_{r \text{ terms}}\underbrace{(N^{m-1} \cdots N^{m-1})}_{r \text{ terms}} \cdots \underbrace{(N \cdots N)}_{r \text{ terms}} \\ &= H^{1+2+\cdots+(d-1)}(N^{1+2+\cdots+m})^r \\ &= H^{d(d-1)/2}N^{rm(m+1)/2}. \end{aligned}$$

Invoking Lemma 2.2, we may compute a nonzero vector $w \in L$ such that

$$\|w\| \leqslant 2^{(d-1)/4} H^{(d-1)/2} N^{rm(m+1)/2d}$$

in time $O(d^{5+\epsilon}(\lg B)^{2+\epsilon})$. Note that this time bound certainly dominates the cost of computing the vectors v_i themselves, as the $\tilde{f}_i(y)$ may be computed by starting with $\tilde{f}_0(y) = N^m$, and then successively multiplying by $P+Hy$ and occasionally dividing by N. The hypotheses (2.1) and (2.2) imply that

$$\lg B \ll d + (\tfrac{d}{r}+1)\lg N = \left(\tfrac{r}{\lg N} + 1 + \tfrac{r}{d}\right)\tfrac{d}{r}\lg N \leqslant \left(2+\tfrac{1}{m}\right)\tfrac{d}{r}\lg N \ll \tfrac{d}{r}\lg N,$$

so the cost estimate $O(d^{5+\epsilon}(\lg B)^{2+\epsilon})$ simplifies to (2.5).

The vector w corresponds to a nonzero polynomial $\tilde{h}(y) = \tilde{h}_0 + \cdots + \tilde{h}_{d-1}y^{d-1}$ in the $\mathbb{Z}$-span of the $\tilde{f}_i(y)$. Applying the Cauchy–Schwartz inequality to the vectors $w = (\tilde{h}_0, \ldots, \tilde{h}_{d-1})$ and $(1, \ldots, 1)$ yields

$$|\tilde{h}_0| + \cdots + |\tilde{h}_{d-1}| \leqslant d^{1/2}\,\|w\| < d^{1/2}\,2^{(d-1)/4} H^{(d-1)/2} N^{rm(m+1)/2d}.$$

Moreover, each $\tilde{h}_j$ is divisible by H^j, so we obtain in turn a polynomial $h(x) := \tilde{h}(x/H) \in \mathbb{Z}[x]_d$ in the $\mathbb{Z}$-span of the $f_i(x)$. Since $h(x) = h_0 + \cdots + h_{d-1}x^{d-1}$ with $h_j = \tilde{h}_j/H^j$ for each j, the estimate (2.6) follows immediately. □

Next we show that any r-power divisor that is sufficiently close to P corresponds to a root of $h(x)$.

Proposition 2.4 *Let N, r, m, d, P and H be positive integers satisfying (2.1), (2.2) and (2.3), and let $h(x) \in \mathbb{Z}[x]_d$ be as in Proposition 2.3. Suppose additionally that (2.4) holds, and that p is an integer in the interval $P - H \leqslant p \leqslant P + H$ such that $p^r \mid N$. Then $x_0 := p - P$ is a root of $h(x)$.*

Proof We claim that $h(x_0)$ is divisible by p^{rm}. Since $h(x)$ is a $\mathbb{Z}$-linear combination of the $f_i(x)$ (where $f_i(x)$ is defined as in Proposition 2.3), it is enough to prove that $p^{rm} \mid f_i(x_0)$ for all i. For the case $0 \leqslant i < rm$, we have $f_i(x_0) = N^{m-\lfloor i/r\rfloor}p^i$. Since $p^r \mid N$, we have $p^{r(m-\lfloor i/r\rfloor)}p^i \mid f_i(x_0)$, and this implies that $p^{rm} \mid f_i(x_0)$ because $r\lfloor i/r\rfloor \leqslant i$. For the case $i \geqslant rm$ we have simply $f_i(x_0) = p^i$, which is certainly divisible by p^{rm}.

On the other hand, the assumption $-H \leqslant x_0 \leqslant H$ together with (2.6) and (2.4) implies that

$$|h(x_0)| \leqslant |h_0| + \cdots + |h_{d-1}|H^{d-1} < (P-H)^{rm} \leqslant p^{rm}.$$

Since $h(x_0)$ is divisible by p^{rm}, this forces $h(x_0) = 0$. □

We may now complete the proof of the main theorem of this section.

Proof of Theorem 2.1 We first invoke Proposition 2.3, with inputs N, r, m, d, P and H, to find a polynomial $h(x)$ satisfying (2.6). According to Proposition 2.4, we may then construct a list of candidates for p by finding all integer roots of $h(x)$, which we do via Theorem 1.3.

To estimate the complexity of the root-finding step, recall from the proof of Proposition 2.4 that $|h_0| + \cdots + |h_{d-1}|H^{d-1} < (P-H)^{rm}$, so certainly $|h_j| < (P-H)^{rm}$ for all j, and we obtain

$$\|h\|_\infty < (P-H)^{rm} < P^{rm} \leqslant N^m \leqslant N^{d/r}.$$

Therefore in Theorem 1.3 we may take $n := d$ and $b := \lceil \lg(N^{d/r}) \rceil = \lceil \frac{d}{r} \lg N \rceil$. Note that the hypothesis $b \geqslant n$ is satisfied due to (2.1). The root-finding complexity is thus

$$O(d^{2+\epsilon}(\tfrac{d}{r} \lg N)^{1+\epsilon}) = O(d^{3+\epsilon}(\tfrac{1}{r} \lg N)^{1+\epsilon}),$$

which is negligible compared to the main bound (2.5). Finally, we must check each candidate for p to ensure that $p^r \mid N$, which again requires negligible time. □

3 Proof of the main theorem

We now consider the problem of searching for all integers p such that $p^r \mid N$ in an interval, say $T \leqslant p \leqslant T'$, that is too large to be handled by a single application of Theorem 2.1. Given N, r, T and T', our strategy will be to choose parameters d, m and H, and then apply Theorem 2.1 to a sequence of subintervals of the form $P - H \leqslant p \leqslant P + H$ that cover the target interval $T \leqslant p \leqslant T'$. The overall running time will depend mainly on the number of subintervals, so our goal is to make H as large as possible. On the other hand, to ensure that the hypothesis (2.4) of Theorem 2.1 is satisfied, we also require that $H < \tilde{H}$ where

$$\tilde{H} := \frac{1}{d^{1/(d-1)}2^{1/2}} \cdot \frac{T^{2rm/(d-1)}}{N^{rm(m+1)/d(d-1)}} > 0. \tag{3.1}$$

The key issue is therefore to choose d and m to maximise $\tilde{H}$. For large d and m, the magnitude of $\tilde{H}$ depends more or less on the ratio m/d; in fact, one finds that $\tilde{H}$ is maximised when $m/d \approx \lg T / \lg N$. The following result gives a simple formula for m (as a function of d) that is close to the optimal choice, and a corresponding explicit lower bound for $\tilde{H}$.

Lemma 3.1 *Let N, r, d and T be positive integers such that $d \geqslant 2$ and $T \leqslant N^{1/r}$. Let*

$$m := \left\lfloor \frac{(d-1)\lg T}{\lg N} \right\rfloor, \tag{3.2}$$

and let $\tilde{H}$ be defined as in (3.1). Then

$$\tilde{H} > \tfrac{1}{3} N^{\theta^2/r \,-\, 1/(d-1)},$$

where

$$\theta := \frac{r \lg T}{\lg N} \in [0, 1] \qquad (\text{so that } T = N^{\theta/r}). \tag{3.3}$$

Proof The definition of m implies that $(d-1)\frac{\theta}{r} - 1 < m \leqslant (d-1)\frac{\theta}{r}$, so we may write

$$\frac{m}{d-1} = \frac{\theta}{r} - \delta \qquad \text{for some } \delta \in [0, \tfrac{1}{d-1}).$$

It is easy to check that $d^{1/(d-1)}2^{1/2} < 3$ for all $d \geqslant 2$, so we find that

$$\tilde{H} > \tfrac{1}{3} N^{2\theta m/(d-1) - rm(m+1)/d(d-1)}.$$

Continuing to estimate the exponent in this inequality, we obtain

$$\begin{aligned}\frac{2\theta m}{d-1}-\frac{rm(m+1)}{d(d-1)} &> \frac{2\theta m}{d-1}-\frac{rm(m+1)}{(d-1)^2}\\ &= \frac{2\theta m}{d-1}-\frac{rm^2}{(d-1)^2}-\frac{rm}{(d-1)^2}\\ &= 2\theta(\tfrac{\theta}{r}-\delta)-r(\tfrac{\theta}{r}-\delta)^2-\frac{r(\frac{\theta}{r}-\delta)}{d-1}\\ &= \frac{\theta^2}{r}+r\delta\left(\frac{1}{d-1}-\delta\right)-\frac{\theta}{d-1}\\ &\geqslant \frac{\theta^2}{r}-\frac{1}{d-1},\end{aligned}$$

where the last line follows from the inequalities $0 \leqslant \delta < \frac{1}{d-1}$ and $0 \leqslant \theta \leqslant 1$. □

We may now estimate the time required to search a given interval $T \leqslant p \leqslant T'$.

Proposition 3.2 *There is an explicit deterministic algorithm with the following properties. It takes as input positive integers N, r, T and T' such that (2.1) holds (i.e., $r \leqslant \lg N$) and such that*

$$4^{\sqrt{(\lg N)/r}} \leqslant T < T' \leqslant N^{1/r}. \tag{3.4}$$

Its output is a list of all integers p in the interval $T \leqslant p \leqslant T'$ such that $p^r \mid N$. Its running time is

$$O\left(\left(\frac{T'-T}{T}\cdot N^{\theta(1-\theta)/r}+1\right)\frac{(\lg N)^{9+\epsilon}}{r^2}\right),$$

where θ is defined as in (3.3).

Proof Set

$$d := \lceil \lg N \rceil + 1$$

and define m as in (3.2). Equivalently, m is the largest integer such that $N^m \leqslant T^{d-1}$. Note that $d \geqslant 2$ (since $N \geqslant 2^r \geqslant 2$) and $m \geqslant \lfloor \lg T \rfloor \geqslant 2$ (since $T \geqslant 4^{\sqrt{1}} = 4$). Since $\lg(T^{d-1}) \leqslant d \lg T \ll (\lg N)^2$, we may clearly compute d and m in time $O((\lg N)^{2+\epsilon})$. Also, the assumption $T \leqslant N^{1/r}$ implies that $m \leqslant (d-1)/r \leqslant d/r$, so (2.2) holds.

Let $\tilde{H}$ be defined as in (3.1). Since $d \geqslant \lg N + 1$, Lemma 3.1 implies that

$$\tilde{H} > \tfrac{1}{3}N^{\theta^2/r}N^{-1/\lg N} = \tfrac{1}{6}N^{\theta^2/r}.$$

Moreover, (3.4) implies that $\theta \geqslant 2\sqrt{r/\lg N}$, so we have $N^{\theta^2/r} \geqslant N^{4/\lg N} = 16$ and hence $\tilde{H} > 16/6 > 2$.

Let H be the largest integer less than $\tilde{H}$, i.e., $H := \lceil \tilde{H} \rceil - 1$. Then $2 \leqslant H < \tilde{H}$, and moreover, since $\tilde{H} > 2$, we also have

$$H \geqslant \tilde{H}/2 > \tfrac{1}{12}N^{\theta^2/r}.$$

We may compute H by first approximating the $d(d-1)$-th root of the rational number

$$\tilde{H}^{d(d-1)} = \frac{T^{2drm}}{d^d 2^{d(d-1)/2} N^{rm(m+1)}},$$

and then taking $d(d-1)$-th powers of nearby integers to find the correct value. The numerator has bit size at most $O(drm \lg T) = O(d^2 \lg N) = O(\lg^3 N)$, and the denominator also has bit size at most

$$O(d \lg d + d^2 + rm^2 \lg N) = O(d^2 + dm \lg N) = O(d^2 \lg N) = O(\lg^3 N),$$

so this can all be done in time $O((\lg N)^{3+\epsilon})$.

We now apply Theorem 2.1 with the parameters N, r, d, m, H, and with successively $P = T+H, P = T+3H$, and so on, stopping when the interval $[T, T']$ has been exhausted by the subintervals $[P-H, P+H]$. The hypotheses (2.1), (2.2) and (2.3) have already been checked above, and (2.4) follows from our choice of $H < \tilde{H}$ because $P - H \geqslant T$. The number of subintervals is at most

$$\left\lceil \frac{T'-T}{2H} \right\rceil \leqslant \frac{T'-T}{\frac{1}{6}N^{\theta^2/r}} + 1 = \frac{6(T'-T)}{T} \cdot N^{\theta(1-\theta)/r} + 1.$$

Finally, since $d \ll \lg N$, the cost of each invocation of Theorem 2.1 is

$$O\big(d^{7+\epsilon}(\tfrac{1}{r}\lg N)^{2+\epsilon}\big) = O\left(\frac{(\lg N)^{9+\epsilon}}{r^2}\right).$$

□

Remark 3.3 A slightly better choice for d is to take $d \approx \theta \lg N$, but this complicates the analysis and only improves the main result by a constant factor.

Finally we may prove the main theorem. Recall that we are given as input positive integers $N \geqslant 2$ and $r \leqslant \lg N$, and we wish to find *all* positive integers p such that $p^r \mid N$. Such divisors p must clearly lie in $[1, N^{1/r}]$.

Proof of Theorem 1.1 Let

$$k := \left\lceil 2\sqrt{\lceil \lg N \rceil / r} \right\rceil.$$

We first check all $p = 2, 3, \ldots, 2^k$ by brute force, i.e., testing directly whether $p^r \mid N$. Note that k may certainly be computed in time $O((\lg N)^{1+\epsilon})$. To estimate the cost of checking up to 2^k, observe that

$$k \leqslant 2\sqrt{(\lg N)/r + 1} + 1.$$

Let $C > 0$ be an absolute constant such that $2\sqrt{x+1} + 1 \leqslant x/4 + C$ for all $x \geqslant 1$; it follows that $k \leqslant (\lg N)/4r + C$, and hence that $2^k \ll N^{1/4r}$. The cost of checking up to 2^k is therefore $O(N^{1/4r}(\lg N)^{1+\epsilon})$, which is negligible compared to (1.1).

We now apply Proposition 3.2 to the intervals $[2^k, 2^{k+1}]$, $[2^{k+1}, 2^{k+2}]$, and so on until we reach $N^{1/r}$, taking the last interval to be $[2^j, \lfloor N^{1/r} \rfloor]$ for suitable j. Since $k \geqslant 2\sqrt{(\lg N)/r}$, the precondition (3.4) is satisfied. For each interval we have $(T'-T)/T = O(1)$, and since $\theta \in [0, 1]$ we have

$$\theta(1-\theta) \leqslant \frac{1}{4}.$$

Therefore the cost of searching each interval is

$$O\left((N^{1/4r} + 1) \cdot \frac{(\lg N)^{9+\epsilon}}{r^2}\right) = O\left(N^{1/4r} \cdot \frac{(\lg N)^{9+\epsilon}}{r^2}\right).$$

Finally, the number of intervals is at most $\lceil \lg(N^{1/r}) \rceil = O(\frac{1}{r} \lg N)$. □

Remark 3.4 The use of dyadic intervals in the above proof was only for convenience; the same argument would work with intervals $[B^j, B^{j+1}]$ for any fixed $B > 1$.

Remark 3.5 The expression $N^{\theta(1-\theta)/r}$ achieves its maximum value $N^{1/4r}$ at the point $\theta = 1/2$. This justifies the claim made in the introduction that the factors p^r that are “hardest” to find are those for which $p \sim N^{1/2r}$.

Remark 3.6 A more careful analysis, taking into account the fact that $N^{\theta(1-\theta)/r}$ is much smaller than $N^{1/4r}$ for most values of $\theta \in [0, 1]$, shows that the bound (1.1) can be improved by a factor of $O((\frac{1}{r} \lg N)^{1/2})$. Let us briefly explain this calculation. The main contribution to the cost estimate in the above proof is the number of subintervals, i.e., the sum of the values of $N^{\theta(1-\theta)/r}$ over the various dyadic intervals. It can be shown that this sum is essentially a Riemann sum approximating the integral

$$\frac{\log N}{r} \int_0^1 N^{\theta(1-\theta)/r} d\theta.$$

The argument in the proof of Theorem 1.1 amounted to estimating this integral via the trivial bound $\int_0^1 N^{\theta(1-\theta)/r} d\theta \leqslant \int_0^1 N^{1/4r} d\theta = N^{1/4r}$. A better estimate is obtained by recognising the integrand as a truncated Gaussian function, i.e.,

$$\begin{aligned}\int_0^1 N^{\theta(1-\theta)/r} d\theta &= \int_{-1/2}^{1/2} N^{(1/4-\alpha^2)/r} d\alpha \\ &\leqslant N^{1/4r} \int_{-\infty}^{\infty} N^{-\alpha^2/r} d\alpha = \left(\frac{\pi r}{\log N}\right)^{1/2} N^{1/4r}.\end{aligned}$$

An interesting question is whether this $(\lg N)^{1/2}$ factor is somehow equivalent to the $(\lg N)^{1/2}$ factor appearing in [2, §6]; we have not checked this in detail.

Acknowledgements
The authors would like to thank Joris van der Hoeven for helpful discussions on the root finding problem.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability Statement Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Author details
[1]School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia, [2]SBA Research, Floragasse 7, 1040 Vienna, Austria.

Appendix A. Deterministic root finding

In this section we prove Theorem 1.3. Our root-finding procedure consists of two parts. In the first part, we discuss how to deterministically find all integer roots of a *squarefree* polynomial $f \in \mathbb{Z}[x]$. We mainly follow the approach of Loos [11], but we obtain better complexity bounds by employing faster algorithms for the underlying arithmetic. In the second part, we explain how to reduce the general case to the squarefree case. The reduction depends on computing GCDs in $\mathbb{Z}[x]$; for this purpose we present a rigorous, deterministic variant of the “heuristic GCD” algorithm of Char, Geddes and Gonnet [5].

A.1. Some preliminary estimates

For $f, g \in \mathbb{Z}[x]$, let $\operatorname{res}(f, g) \in \mathbb{Z}$ denote the resultant of f and g.

Lemma A.1 *Let $f, g \in \mathbb{Z}[x]$ be nonzero polynomials, and let $n := \deg f$, $m := \deg g$. Then*

$$|\operatorname{res}(f, g)| \leqslant (n+1)^{m/2}(m+1)^{n/2} \, \|f\|_\infty^m \, \|g\|_\infty^n .$$

Proof See [19, Thm. 6.23]. (The proof uses Hadamard's bound to estimate the determinant of the Sylvester matrix associated to f and g.) □

Lemma A.2 (Mignotte's factor bound) *Let $f, g \in \mathbb{Z}[x]$ be nonzero polynomials, and let $n := \deg f$, $m := \deg g$. If g divides f in $\mathbb{Z}[x]$ then*

$$\|g\|_\infty \leqslant (n+1)^{1/2} \, 2^m \, \|f\|_\infty .$$

Proof See [19, Cor. 6.33(ii)]. (The proof relies on Landau's inequality for the Mahler measure of a polynomial.) □

Lemma A.3 *For all $X \geqslant 2$ we have*

$$\sum_{p \leqslant X} \lg p > X/3$$

(where the sum is taken over primes).

Proof Let $\vartheta(X) := \sum_{p \leqslant X} \log p$ denote the usual Chebyshev weighted prime counting function. The claim is that $\vartheta(X)/X > \frac{1}{3} \log 2$ (≈ 0.231) for all $X \geqslant 2$. For $X \geqslant 101$ this follows from [15, Thm. 10], which states that $\vartheta(X)/X > 0.84$ for $X \geqslant 101$. For $2 \leqslant X < 101$ the claim may be checked directly, for example by inspecting the graph of $\vartheta(X)/X$ in the reader's favourite computer algebra system. □

A.2. The squarefree case

The core idea of Loos' algorithm is the following well-known p-adic Hensel lifting strategy.

Proposition A.4 *Let p be a prime, let k be a positive integer, and let $f \in (\mathbb{Z}/p^k\mathbb{Z})[x]$ be a polynomial of degree $n \geqslant 1$. Let $u \in \{0, \ldots, p-1\}$, and suppose that*

$$f(u) \equiv 0 \pmod{p}, \qquad f'(u) \not\equiv 0 \pmod{p}.$$

Then there exists a unique $v \in \{0, \ldots, p^k - 1\}$ such that

$$v \equiv u \pmod{p}, \qquad f(v) \equiv 0 \pmod{p^k}.$$

Given f and u as input, we may compute v in time

$$O(n \lg(p^k)^{1+\epsilon}).$$

Proof We argue by induction on k. If $k = 1$, we simply take $v = u$. Now assume that $k \geqslant 2$ and set $\ell := \lceil k/2 \rceil < k$. By induction there exists a unique $w \in \{0, \ldots, p^\ell - 1\}$ such that $w \equiv u \pmod{p}$ and $f(w) \equiv 0 \pmod{p^\ell}$.

We first establish uniqueness of v. Suppose that v has the desired properties, i.e., $v \equiv u \pmod{p}$ and $f(v) \equiv 0 \pmod{p^k}$. By the uniqueness of w, we must have $v \equiv w \pmod{p^\ell}$, say $v = w + p^\ell t$ for some $t \in \{0, \dots, p^{k-\ell} - 1\}$. Expanding f around w, we find that

$$f(w+x) = f(w) + xf'(w) + x^2 g(x)$$

for some $g \in (\mathbb{Z}/p^k\mathbb{Z})[x]$. Substituting $x = p^\ell t$, and using the fact that $p^{2\ell} \equiv 0 \pmod{p^k}$, we deduce that $0 \equiv f(w) + p^\ell t f'(w) \pmod{p^k}$. Since $f'(w) \equiv f'(u) \not\equiv 0 \pmod{p}$, we may solve for t to obtain

$$t \equiv \frac{-f(w)/p^\ell}{f'(w)} \pmod{p^{k-\ell}}.$$

This establishes uniqueness of $t \pmod{p^{k-\ell}}$, and hence of $v \pmod{p^k}$. Moreover, the same calculation gives an explicit formula for v, proving existence.

To prove the complexity bound, suppose that we have already computed w and that we wish to lift to v. We first apply Horner's rule to compute $f(w)$ and $f'(w)$ using $O(n)$ arithmetic operations in $\mathbb{Z}/p^k\mathbb{Z}$. Each such operation requires time $O(\lg(p^k)^{1+\epsilon})$. Similarly, we may invert $f'(w)$, and hence compute t and v, in time $O(\lg(p^k)^{1+\epsilon})$. Therefore, the time required to deduce v from w is $O(n \lg(p^k)^{1+\epsilon})$. The contributions from subsequent recursion levels form a geometric series, so the total cost of computing v from u is also $O(n \lg(p^k)^{1+\epsilon})$. □

The next result shows how to find a reasonably small prime p for which the p-adic lifting strategy is guaranteed to succeed.

Proposition A.5 *Let $b \geqslant n \geqslant 1$ be integers, and let $f \in \mathbb{Z}[x]$ be a squarefree polynomial of degree n such that $\|f\|_\infty \leqslant 2^b$. Then in time*

$$O(n^{2+\epsilon} b^{1+\epsilon})$$

we may find a prime number

$$p \leqslant 6nb + 6n \lg n$$

such that the reduction of f modulo p is nonzero and squarefree in $(\mathbb{Z}/p\mathbb{Z})[x]$.

Proof Since f is squarefree, the resultant $D := \operatorname{res}(f, f')$ is nonzero. Our goal is to find a prime p such that $p \nmid D$.

First, by Lemma A.1 we have

$$|D| \leqslant (n+1)^{(n-1)/2} n^{n/2} \|f\|_\infty^{n-1} \|f'\|_\infty^n.$$

One easily checks that $(n+1)^{n-1} \leqslant n^n$ for all $n \geqslant 1$. Since $\|f'\|_\infty \leqslant n \|f\|_\infty$, we obtain

$$|D| \leqslant n^{2n} 2^{2nb}. \tag{A.1}$$

On the other hand, let $X := 6bn + 6n \lg n$. If D is divisible by all primes $p \leqslant X$, then D is divisible by their product, so

$$\lg |D| \geqslant \sum_{p \leqslant X} \lg p > X/3 = 2nb + 2n \lg n$$

by Lemma A.3. This contradicts (A.1), so we conclude that there must exist a prime $p \leqslant X$ such that $p \nmid D$. To actually find such a prime, we run the following algorithm.

Step 1 (list primes). Make a list of all primes $p \leqslant Y$ for $Y := 6nb + 6n\lceil \lg n \rceil = O(nb)$. Using the sieve of Eratosthenes, this requires time $O(Y^{1+\epsilon}) = O(n^{1+\epsilon} b^{1+\epsilon})$.

Step 2 (reduce f modulo primes). Let $f_p \in (\mathbb{Z}/p\mathbb{Z})[x]$ denote the reduction of f modulo p. We compute f_p for all $p \leqslant Y$ by applying a fast simultaneous modular reduction algorithm [19, Thm. 10.24] (i.e., using a remainder tree) to each coefficient of f. The bit size of the product of the primes is $O(\vartheta(Y)) = O(Y) = O(nb)$, and the number of primes is certainly $O(nb)$, so the cost per coefficient is $O(n^{1+\epsilon}b^{1+\epsilon})$. The total cost over all coefficients is therefore $O(n^{2+\epsilon}b^{1+\epsilon})$.

Step 3 (compute GCDs). For each $p \leqslant Y$, we compute $\gcd(f_p, f_p') \in (\mathbb{Z}/p\mathbb{Z})[x]$ using a quasilinear time GCD algorithm [19, Cor. 11.9]. For each prime this requires $O(n^{1+\epsilon})$ ring operations in $\mathbb{Z}/p\mathbb{Z}$, and each ring operation costs $O((\lg p)^{1+\epsilon})$ bit operations. The hypothesis $n \leqslant b$ implies that $\lg p = O(\lg(nb)) = O(\lg b)$, so the cost of computing the GCD is $O(n^{1+\epsilon}b^{\epsilon})$ bit operations. The total cost over all $O(nb)$ primes is therefore $O(n^{2+\epsilon}b^{1+\epsilon})$.

Finally, we return the least prime p for which $f_p \neq 0$ and $\gcd(f_p, f_p') = 1$. As shown above, such a prime exists and satisfies $p \leqslant X$. □

We now give a deterministic root-finding algorithm for the squarefree case.

Proposition A.6 *Let $b \geqslant n \geqslant 1$ be integers, and let $f \in \mathbb{Z}[x]$ be a squarefree polynomial of degree n such that $\|f\|_\infty \leqslant 2^b$. Then we may find all integer roots of f in time*

$$O(n^{2+\epsilon}b^{1+\epsilon}).$$

Proof As above, let $f_p \in (\mathbb{Z}/p\mathbb{Z})[x]$ denote the reduction of f modulo p. We first invoke Proposition A.5 to find a prime $p = O(nb)$ such that f_p is nonzero and squarefree. Then we perform the following steps.

Step 1 (find roots mod p). Compute the roots of f_p in $\mathbb{Z}/p\mathbb{Z}$ by brute force, i.e., by evaluating $f_p(i)$ for $i = 0, \ldots, p-1$. Note that the integer roots of f correspond to *distinct* roots of f_p, thanks to the squarefreeness of f_p. Each $f_p(i)$ may be evaluated in time $O(n^{1+\epsilon}b^{\epsilon})$, so the cost of this step is $O(pn^{1+\epsilon}b^{\epsilon}) = O(n^{2+\epsilon}b^{1+\epsilon})$.

Step 2 (find roots mod p^k). Let $\bar{f} \in (\mathbb{Z}/p^k\mathbb{Z})[x]$ be the reduction of f modulo p^k, where k is chosen to be the smallest integer such that

$$p^k > (n+1)^{1/2}\, 2^{n+b+1}. \tag{A.2}$$

Applying Proposition A.4 to $\bar{f}$, we lift each of the roots of f_p found in Step 1 to a root of $\bar{f}$. The uniqueness claim in Proposition A.4 implies that the resulting set of lifted roots in $\mathbb{Z}/p^k\mathbb{Z}$ includes the reductions modulo p^k of all of the actual integer roots of f. To estimate the complexity, observe that $p^k \leqslant p(n+1)^{1/2}\, 2^{n+b+1}$, so

$$\lg(p^k) = O(\lg p + \lg(n+1) + n + b) = O(b).$$

The cost of lifting each root is therefore $O(n \lg(p^k)^{1+\epsilon}) = O(nb^{1+\epsilon})$, and the total cost of this step is $O(n^2 b^{1+\epsilon})$.

Step 3 (check roots in $\mathbb{Z}$). For each root $\bar{r} \in \mathbb{Z}/p^k\mathbb{Z}$ of $\bar{f}$ found in Step 2, we determine whether it arises from a genuine integer root of f as follows. We first lift $\bar{r}$ to a candidate root $r^* \in \mathbb{Z}$ satisfying $r^* \equiv \bar{r} \pmod{p^k}$ and $r^* \in [-\frac{1}{2}p^k, \frac{1}{2}p^k)$. We next divide $\bar{f}$ by $x - \bar{r}$ to obtain a polynomial $\bar{g} \in (\mathbb{Z}/p^k\mathbb{Z})[x]$ such that $\bar{f}(x) = (x - \bar{r})\bar{g}(x)$, and we lift $\bar{g}$ to a polynomial $g^* \in \mathbb{Z}[x]$ satisfying $g^* \equiv \bar{g} \pmod{p^k}$ and whose coefficients also all lie in $[-\frac{1}{2}p^k, \frac{1}{2}p^k)$. We then multiply $x - r^*$ by $g^*(x)$ (in $\mathbb{Z}[x]$) and check whether we obtain

f. If so, then $f(r^*) = 0$, so r^* must be the integer root corresponding to $\bar{r}$. Otherwise, as we will see in the next paragraph, this $\bar{r}$ does not correspond to any integer root and we may ignore it. This procedure requires $O(n)$ operations on integers of $O(b)$ bits, i.e., $O(nb^{1+\epsilon})$ bit operations, so the total cost over all roots is $O(n^2b^{1+\epsilon})$.

We now prove that the procedure described above does in fact find all integer roots. (The following argument is adapted from [19, §15.6].) Let $r \in \mathbb{Z}$ be a root of f. Then $|r| \leqslant \|f\|_\infty \leqslant 2^b$ (as r divides the constant term of f), and f factors as $f(x) = (x - r)g(x)$ for some $g \in \mathbb{Z}[x]$ satisfying $\|g\|_\infty \leqslant (n + 1)^{1/2}\, 2^{n+b}$ (by Lemma A.2). In particular, (A.2) ensures that $|r| < p^k/2$ and $\|g\|_\infty < p^k/2$. Let $\bar{r} \in \mathbb{Z}/p^k\mathbb{Z}$ be the root of $\bar{f}$ corresponding to r, and let $r^* \in \mathbb{Z}$ and $g^* \in \mathbb{Z}[x]$ be the quantities computed in Step 3 for this $\bar{r}$. Then $r^* \equiv \bar{r} \equiv r \pmod{p^k}$, so we must have $r^* = r$, since both sides lie in $[-\frac{1}{2}p^k, \frac{1}{2}p^k)$. Similarly, we have

$$g^*(x) \equiv \bar{g}(x) = \bar{f}(x)/(x - \bar{r}) \equiv f(x)/(x - r) = g(x) \pmod{p^k},$$

so again we must have $g^* = g$ as the coefficients on both sides lie in $[-\frac{1}{2}p^k, \frac{1}{2}p^k)$. Therefore $(x - r^*)g^*(x) = f(x)$, and the procedure does indeed recover r. □

Remark A.7 Loos [11] imposes the additional requirement that p should not divide the leading coefficient of f, to ensure that $\deg f_p = \deg f$. This is because he is searching for *rational* roots, not just integral roots. Our algorithm may also be easily adapted to this case.

Remark A.8 An interesting question is whether the complexity bound in Proposition A.6 can be improved to quasilinear, i.e., to $O(n^{1+\epsilon}b^{1+\epsilon})$ bit operations. There are two main obstructions to this.

First, although $f_p \in (\mathbb{Z}/p\mathbb{Z})[x]$ is squarefree for almost all primes p, it is difficult to predict in advance for which p this will occur. Consequently, in the proof of Proposition A.5 we were forced to test *every* prime up to $O(nb)$. If we allow probabilistic algorithms, then we can find a suitable prime with high probability by randomly selecting p in the range $2 \leqslant p \leqslant X'$ for some $X' = O(nb)$. This allows us to find a suitable p in *expected* quasilinear time. The complexity of Steps 1 and 2 in Proposition A.6, i.e., finding the roots modulo p and lifting them to $\mathbb{Z}/p^k\mathbb{Z}$, can also be improved to (deterministic) quasilinear time by means of fast multipoint evaluation techniques. The resulting algorithm is quite similar to the root finding algorithm presented in [19, Thm. 15.21].

The second obstruction concerns Step 3 of Proposition A.6, namely, checking which of the candidate integer roots are in fact roots of f. We do not know how to carry out this step rigorously in quasilinear time, even allowing randomised algorithms. A similar issue occurs in [19, Thm. 15.21], where the last term of the given complexity bound corresponds in our notation to $O(n^{2+\epsilon}b^{1+\epsilon})$. In the discussion following that theorem, von zur Gathen and Gerhard suggest testing the candidate roots modulo a small prime (different to p) as a way to quickly rule out incorrect candidates. This idea can be turned into a "Monte Carlo" algorithm: one would randomly choose a small prime q, compute $f(r^*) \pmod{q}$ for all candidate roots r^*, and declare the ones for which $f(r^*) \equiv 0 \pmod{q}$ to be the true roots. We suspect that in this way one can obtain a quasilinear expected running time with an exponentially small probability of failure, but we have not checked the details.

A.3. The general case

In order to prove Theorem 1.3, we must first discuss the computation of GCDs in $\mathbb{Z}[x]$.

Let $f, g \in \mathbb{Z}[x]$ and let $h := \gcd(f, g)$. The idea of the "heuristic GCD" algorithm [5] is to use an *integer* GCD algorithm to compute $\gcd(f(N), g(N))$ for some choice of evaluation point $N \in \mathbb{Z}$. If we are lucky, then $\gcd(f(N), g(N))$ will actually be equal to $h(N)$, and we may simply read off the coefficients of $h(x)$ from $h(N)$, provided that N is not too small. However, it is possible for $\gcd(f(N), g(N))$ to contain extraneous factors unrelated to $h(x)$. Usually these extraneous factors are small but in rare circumstances they can be very large. The algorithm can be made to tolerate extraneous factors up to a given size by taking larger values of N, at the expense of running more slowly. In practice, one usually takes a fairly small value of N, accepting a small chance of failure in order to get a fast algorithm. In the next result we work at the other extreme, taking N so large that the algorithm is guaranteed to work in all cases. We thereby obtain a GCD algorithm that is deterministic and completely rigorous (although unfortunately quite slow in practice).

Proposition A.9 *Let $b \geqslant n \geqslant 1$ be integers. Let $f, g \in \mathbb{Z}[x]$ be nonzero polynomials such that $\deg f, \deg g \leqslant n$ and $\|f\|_\infty, \|g\|_\infty \leqslant 2^b$. Assume that at least one of f and g is primitive. Define*

$$h := \gcd(f, g) \in \mathbb{Z}[x], \qquad \tilde{f} := f/h \in \mathbb{Z}[x], \qquad \tilde{g} := g/h \in \mathbb{Z}[x].$$

Then

$$\|h\|_\infty, \|\tilde{f}\|_\infty, \|\tilde{g}\|_\infty \leqslant (n+1)^{1/2}\, 2^{n+b}, \tag{A.3}$$

and given f and g as input, we may compute $h, \tilde{f}$ and $\tilde{g}$ in time

$$O(n^{2+\epsilon} b^{1+\epsilon}).$$

Proof The inequalities (A.3) follow immediately from Lemma A.2, as $\tilde{f}$ and $\tilde{g}$ are divisors of f and g respectively, and h is a divisor of both.

For any integer N we have $f(N) = \tilde{f}(N)h(N)$ and $g(N) = \tilde{g}(N)h(N)$, so

$$\gcd(f(N), g(N)) = \delta(N) \cdot h(N) \qquad \text{where } \delta(N) := \gcd(\tilde{f}(N), \tilde{g}(N)).$$

Writing $h(x) = h_0 + h_1 x + \cdots + h_n x^n$, this becomes

$$\gcd(f(N), g(N)) = \delta(N)h_0 + \delta(N)h_1 N + \cdots + \delta(N)h_n N^n. \tag{A.4}$$

We may bound the quantities $\delta(N)h_i$ independently of N as follows. (This argument is adapted from [7, Thm. 4].) Since $\tilde{f}$ and $\tilde{g}$ are relatively prime, their resultant $R := \operatorname{res}(\tilde{f}, \tilde{g})$ is nonzero, and there exist polynomials $r, s \in \mathbb{Z}[x]$ such that

$$r(x)\tilde{f}(x) + s(x)\tilde{g}(x) = R.$$

Substituting $x = N$ shows that $\delta(N) \mid R$. Applying Lemma A.1, we obtain

$$|\delta(N)| \leqslant |R| \leqslant (n+1)^n \|\tilde{f}\|_\infty^n \|\tilde{g}\|_\infty^n \leqslant (n+1)^{2n}\, 2^{2n^2+2nb}.$$

Therefore the quantities $\delta(N)h_i$ are bounded by

$$|\delta(N)h_i| \leqslant |\delta(N)|\, \|h\|_\infty \leqslant (n+1)^{2n+\frac{1}{2}}\, 2^{2n^2+2nb+n+b}.$$

We may now describe the actual algorithm for computing $h, \tilde{f}$ and $\tilde{g}$.

Step 1. Set $N := 2^c$ where

$$c := (2n+1)\lceil \lg(n+1) \rceil + 2n^2 + 2nb + n + b + 2.$$

As shown above, $|\delta(N)h_i| \leqslant 2^{c-2}$ for all i. Notice also that $c = O(nb)$.

Step 2. We compute $f(N)$ and $g(N)$, which amounts to concatenating the coefficients of f and g with appropriate zero-padding (or one-padding in the case of negative coefficients). The integers $f(N)$ and $g(N)$ have bit size $O(nc) = O(n^2 b)$, and the concatenation may be performed in linear time, i.e., in time $O(n^2 b)$.

Step 3. We compute $\gcd(f(N), g(N))$ using a quasilinear time GCD algorithm (see for example [18]). This requires time $O((n^2 b)^{1+\epsilon}) = O(n^{2+\epsilon} b^{1+\epsilon})$.

Step 4. We read off the coefficients $\delta(N)h_i$ from (A.4). This is possible thanks to the bound $|\delta(N)h_i| \leqslant 2^{c-2}$, i.e., the coefficients do not "overlap". (In more detail, we may first read off $\delta(N)h_0$ from the lowest c bits, i.e., by reading (A.4) modulo $N = 2^c$. After subtracting off this term, we may read off $\delta(N)h_1$ from the next c bits, and so on.) This requires linear time $O(n^2 b)$.

Step 5. Since we assumed that at least one of f and g is primitive, h is also primitive. We may therefore compute $\delta(N)$ by taking the GCDs of the integers $\delta(N)h_0, \ldots, \delta(N)h_n$. Each pairwise GCD requires time $O(c^{1+\epsilon})$, so the total time required for this step is $O(nc^{1+\epsilon}) = O(n^{2+\epsilon} b^{1+\epsilon})$.

Step 6. We now recover $h(N) = \gcd(f(N), g(N))/\delta(N)$, and then $\tilde{f}(N) = f(N)/h(N)$ and $\tilde{g}(N) = g(N)/h(N)$. Using a quasilinear time integer division algorithm, this requires time $O(n^{2+\epsilon} b^{1+\epsilon})$. Finally, we read off the coefficients of $\tilde{f}$ and $\tilde{g}$ from $\tilde{f}(N)$ and $\tilde{g}(N)$, in a similar manner to Step 4.

□

Remark A.10 To the best of the authors' knowledge, it is not known how to improve the complexity bound in Proposition A.9 to quasilinear without giving up on determinism. Several randomised quasilinear-time algorithms are known. Schönhage [16] analyses a variant of the heuristic GCD algorithm in which the evaluation point is chosen randomly. Another approach is to compute the GCD modulo a collection of randomly chosen small primes [19, Alg. 6.38].

We may now prove our main root-finding result.

Proof of Theorem 1.3 We are given as input $f \in \mathbb{Z}[x]$, not necessarily squarefree, with $\deg f = n$ and $\|f\|_\infty \leqslant 2^b$, where $b \geqslant n \geqslant 1$.

We first compute the GCD of the coefficients of f, and remove this common factor. Clearly this can be done in time $O(n^{1+\epsilon} b^{1+\epsilon})$, and we may subsequently assume that f is primitive.

Let $g := f'$. Then $\deg g \leqslant n$ and $\|g\|_\infty \leqslant n\|f\|_\infty \leqslant 2^{b'}$ where $b' := b + \lceil \lg n \rceil = O(b)$. Applying Proposition A.9, we may compute $\tilde{f} = f/\gcd(f, f') \in \mathbb{Z}[x]$ in time $O(n^{2+\epsilon}(b')^{1+\epsilon}) = O(n^{2+\epsilon} b^{1+\epsilon})$. Then $\tilde{f}$ is squarefree and has the same integer roots as f. Moreover we have $\deg \tilde{f} \leqslant n$ and $\|\tilde{f}\|_\infty \leqslant (n+1)^{1/2} 2^{n+b'} \leqslant 2^{b''}$ where $b'' := n + b' + \lceil \lg(n+1) \rceil = O(b)$.

Finally, we apply Proposition A.6 to $\tilde{f}$. The running time is $O(n^{2+\epsilon}(b'')^{1+\epsilon}) = O(n^{2+\epsilon} b^{1+\epsilon})$. □

Received: 15 September 2022 Accepted: 21 September 2022
Published online: 22 October 2022

References

1. Boneh, D., Durfee, G., Howgrave-Graham, N.: Factoring $N = p^r q$ for large r, Advances in cryptology—CRYPTO '99 (Santa Barbara, CA), Lecture Notes in Comput. Sci., vol. 1666, Springer, Berlin, 1999, pp. 326–337. **MR 1729303**
2. Bernstein, D. J.: Reducing lattice bases to find small-height values of univariate polynomials, Algorithmic number theory: lattices, number fields, curves and cryptography, Math. Sci. Res. Inst. Publ., vol. 44, Cambridge Univ. Press, Cambridge, 2008, pp. 421–446. MR 2467553
3. Booker, A.R., Hiary, G.A., Keating, J.P.: Detecting squarefree numbers. Duke Math. J. **164**(2), 235–275 (2015). (**MR 3306555**)
4. Brent, R.P., Zimmermann, P.: Modern Computer Arithmetic, Cambridge Monographs on Applied and Computational Mathematics, vol. 18. Cambridge University Press, Cambridge (2011).. (**MR 2760886**)
5. Char, B.W., Geddes, K.O., Gonnet, G.H.: GCDHEU: heuristic polynomial GCD algorithm based on integer GCD computation, EUROSAM 84 (Cambridge, 1984), Lecture Notes in Comput. Sci., Vol. 174, Springer, Berlin (1984), pp. 285–296. MR 779134
6. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptol. **10**(4), 233–260 (1997). (**MR 1476612**)
7. Davenport, J., Padget, J.: HEUGCD: how elementary upperbounds generate cheaper data, EUROCAL '85, Vol. 2 (Linz, 1985), Lecture Notes in Comput. Sci., vol. 204, Springer, Berlin, (1985), pp. 18–28. MR 826554
8. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press, Cambridge (2012).. (**MR 2931758**)
9. Harvey, D., Hittmeir, M.: A log-log speedup for exponent one-fifth deterministic integer factorisation, to appear in Math. Comp. **91**, 1367–1379 (2022)
10. Lenstra, A.K., Lenstra, H.W., Jr., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982). (**MR 682664**)
11. Loos, R.: Computing rational zeros of integral polynomials by p-adic expansion. SIAM J. Comput. **12**(2), 286–293 (1983). (**MR 697160**)
12. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley Publishing Company, Reading, MA (1994).. (**MR 1251285 (95f:68082)**)
13. Peralta, R., Okamoto, E.: Faster factoring of integers of a special form. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **79**(4), 489–493 (1996)
14. Pollard, J.M.: Theorems on factorization and primality testing. Proc. Camb. Philos. Soc. **76**, 521–528 (1974). (**MR 0354514 (50 #6992)**)
15. Rosser, J.B., Schoenfeld, L.: Approximate formulas for some functions of prime numbers. Illinois J. Math. **6**, 64–94 (1962). (**MR 0137689**)
16. Schönhage, A.: Probabilistic computation of integer polynomial GCDs. J. Algorithms **9**(3), 365–371 (1988). (**MR 955145**)
17. Strassen, V.: Einige Resultate über Berechnungskomplexität, Jber. Deutsch. Math.-Verein. **78**(1), 1–8. (1976/77) MR 0438807 (55 #11713)
18. Stehlé, D., Zimmermann, P.: A binary recursive gcd algorithm, Algorithmic number theory, Lecture Notes in Comput. Sci., Vol. 3076, Springer, Berlin (2004), pp. 411–425. MR 2138011
19. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra, 3rd edn. Cambridge University Press, Cambridge (2013)