Contemporary Mathematics

Efficiently Detecting Torsion Points and Subtori

J. Maurice Rojas

To Helaman Ferguson

Abstract. Suppose X is the complex zero set of a finite collection of polynomials in $\mathbb{Z}[x_1, ..., x_n]$. We show that deciding whether X contains a point all of whose coordinates are $d^{\underline{\text{th}}}$ roots of unity can be done within $\mathbf{NP}^{\mathbf{NP}}$ (relative to the sparse encoding), under a plausible assumption on primes in arithmetic progression. In particular, our hypothesis can still hold even under certain failures of the Generalized Riemann Hypothesis, such as the presence of Siegel-Landau zeroes. Furthermore, we give a similar **unconditional** complexity upper bound for $n=1$. Finally, letting T be any algebraic subgroup of $(\mathbb{C}^*)^n$ we show that deciding $X \overset{?}{\supseteq} T$ is **coNP**-complete (relative to an even more efficient encoding), unconditionally. We thus obtain new non-trivial families of multivariate polynomial systems where deciding the existence of complex roots can be done unconditionally in the **polynomial hierarchy** — a family of complexity classes lying between **PSPACE** and **P**, intimately connected with the $\mathbf{P} \overset{?}{=} \mathbf{NP}$ Problem. We also discuss a connection to Laurent's solution of Chabauty's Conjecture from arithmetic geometry.

1. Introduction

While the algorithmic complexity of many fundamental problems in algebraic geometry remains unknown, important recent advances have revealed that algebraic geometry and algorithmic complexity are closely and subtly intertwined. For instance, consider the problem of deciding whether a complex algebraic set — specified as the zero set of a collection of multivariate polynomials — is empty or not. This is the **complex feasibility problem**, $\texttt{FEAS}_{\mathbb{C}}$, and we denote its restriction to any family $\mathcal{F}$ of polynomial systems by $\texttt{FEAS}_{\mathbb{C}}(\mathcal{F})$.

Note: The complexity classes we are about to mention are reviewed briefly in Section 3 (see [**Pap95**] for an excellent introductory account).

Department of Mathematics, Texas A&M University TAMU 3368, College Station, Texas 77843-3368, USA. e-mail: `rojas@math.tamu.edu` , web page: `www.math.tamu.edu/~rojas` . Partially supported by NSF individual grant DMS-0211458, NSF CAREER grant DMS-0349309, and Sandia National Laboratories.

Before seminal work of Pascal Koiran [**Koi96**], the only connection known between $\mathtt{FEAS}_{\mathbb{C}}$ and the $\mathbf{P} \overset{?}{=} \mathbf{NP}$ problem was that $\mathtt{FEAS}_{\mathbb{C}}$ is $\mathbf{NP}$-hard, i.e., a polynomial time algorithm for $\mathtt{FEAS}_{\mathbb{C}}$ would imply $\mathbf{P} = \mathbf{NP}$. (The $\mathbf{P} \overset{?}{=} \mathbf{NP}$ problem is the most famous open problem from theoretical computer science and has a vast literature (see, e.g., [**Sma00**] and the references in [**GJ79, Pap95**]).) However, $\mathbf{NP}$-hardness tells us little about what complexity class $\mathtt{FEAS}_{\mathbb{C}}$ actually belongs to, or how quickly we can anticipate solving a given instance of $\mathtt{FEAS}_{\mathbb{C}}$. Koiran's paper [**Koi96**] was the first to show that the truth of the **Generalized Riemann Hypothesis (GRH)** yields the implication $\mathtt{FEAS}_{\mathbb{C}} \not\in \mathbf{P} \Longrightarrow \mathbf{P} \neq \mathbf{NP}$, and [**Roj03**] later showed that this implication could still hold even under certain failures of GRH. Furthermore, the underlying algorithms are entirely different from the usual techniques of commutative algebra (e.g., Gröbner bases and resultants) and thus breathe new life into an old problem.

Here we present algorithms revealing new non-trivial families $\mathcal{F}$ of multivariate polynomial systems where the implication $\mathtt{FEAS}_{\mathbb{C}}(\mathcal{F}) \not\in \mathbf{P} \Longrightarrow \mathbf{P} \neq \mathbf{NP}$ holds **unconditionally**. We also present several examples indicating that the algorithms yielding our main results may be quite practical. In the coming sections, we will detail some of the intricacies behind making such algorithms free from unproved number-theoretic hypotheses. We begin by stating a number-theoretic hypothesis that is demonstrably weaker than GRH. We use $\mathbb{N}$ for the positive integers.

Arithmetic Progression Hypothesis (APH). *There is an absolute constant $C \geq 1$ such that for any $x, M \in \mathbb{N}$ with $x \geq 2^{\log^C M}$, the set $\{1 + kM \mid k \in \{1, \ldots, x\}\}$ contains at least $\frac{x}{\log^C(xM)}$ primes.*

Assumptions even stronger than APH are routinely used, and widely believed, in the cryptology and algorithmic number theory communities (see, e.g., [**Mil76, Mih94, Koi97, Roj01, Hal05**]). In particular, while APH is implied by GRH for the number fields $\{\mathbb{Q}(\omega_M)\}_{M \in \mathbb{N}}$, where ω_M denotes a primitive $M^{\underline{\text{th}}}$ root of unity, APH can still hold under certain failures of the latter hypotheses, e.g., the presence of infinitely many zeroes off the critical line [**Roj03**].

Theorem 1.1. *Suppose $f_1, \ldots, f_k \in \mathbb{Z}[x_1, \ldots, x_n]$, $x := (x_1, \ldots, x_n)$, and $d_1, \ldots, d_n \in \mathbb{N}$. Let* `TorsionPoint` *denote the following problem: Decide whether the system of equations*

$$f_1(x) = \cdots = f_k(x) = x_1^{d_1} - 1 = \cdots = x_n^{d_n} - 1 = 0$$

has a solution in $\mathbb{C}^n$. Also let the **input size** *of the preceding polynomial system be $\left(\sum_{i=1}^k \mathrm{size}(f_i)\right) + \sum_{i=1}^n \mathrm{size}(x^{d_i} - 1)$, where $\mathrm{size}\left(\sum_{i=1}^m c_i x^{a_{i1}} \cdots x_n^{a_{in}}\right) := \sum_{i=1}^m \log\{(|c_i|+2)(a_{i1}+2)\cdots(a_{in}+2)\}$, and let* $\mathtt{TorsionPoint}_1$ *denote the restriction of* `TorsionPoint` *to univariate polynomials. Then*

1. $\mathtt{TorsionPoint} \in \mathbf{AM}$, *assuming APH.*
2. *Unconditionally,* $\mathtt{TorsionPoint}_1 \in \mathbf{NP}^{\mathbf{NP}}$ *and* $\mathtt{TorsionPoint}_1$ *is already* $\mathbf{NP}$*-hard.*
3. *When restricted to* **fixed** n **and** $d_1, \ldots, d_n$, $\mathtt{TorsionPoint} \in \mathbf{P}$ *unconditionally.*

In particular, $\mathtt{TorsionPoint}_1 \not\in \mathbf{P} \Longleftrightarrow \mathbf{P} \neq \mathbf{NP}$ *unconditionally.*

Our notion of input size is quite natural: To put it roughly, $\mathrm{size}(f)$ measures the amount of ink (or memory) one must use to record the monomial term expansion of f. Note that the degree of a polynomial can be exponential in its input size if the polynomial is sparse, e.g., $\text{size}(11z-2xy^{97}z+x^D)=\Theta(\log D)$. (We employ the usual computer science notations $O(\cdot)$ and $\Omega(\cdot)$ to respectively denote upper and lower bounds that are asymptotically true up to a multiplicative constant. When both conditions hold, then one writes $\Theta(\cdot)$.) Thus, in the miraculous event that $\mathbf{P}=\mathbf{NP}$, our algorithm yielding Assertion (2) above has complexity **polynomial** in the bit-sizes of the f_i and the **logarithms** of the d_i — a property **not** present in any earlier algorithm for $\texttt{TorsionPoint}_1$.

Alternatively, Theorem 1.1 tells us that we can try to prove $\mathbf{P}\neq\mathbf{NP}$ by showing that $\texttt{TorsionPoint}_1 \not\in \mathbf{P}$, thus giving another opportunity for algebraic geometry tools for the $\mathbf{P}\overset{?}{=}\mathbf{NP}$ problem (see also [**MS01**] for a different approach via geometric invariant theory). Indeed, should one eventually prove **unconditionally** that `TorsionPoint` lies in the polynomial hierarchy then it would be more profitable to proceed with an attempt to prove $\texttt{TorsionPoint} \not\in \mathbf{P}$ rather than $\texttt{TorsionPoint}_1 \not\in \mathbf{P}$ (since `TorsionPoint` is at least as hard a problem as $\texttt{TorsionPoint}_1$).

Example 1.2 (A Sparse, but Large, Resultant). *Suppose we would like to know if $f(x_1):=c_1+c_2x_1^{a_2}+\cdots+c_{m-1}x_1^{a_{m-1}}+c_mx_1^D$ vanishes at some $M^{\underline{th}}$ root of unity, where $m=\Theta(\log^2 M)$, the c_i are integers of absolute value bounded above by* 10, *and $a_2<\cdots<a_{m-1}<D<M$ are positive integers. The classical resultant for two polynomials in one variable (see, e.g.,* [**GKZ94**]*) then tells us that f vanishes at an $M^{\underline{th}}$ root of unity iff the determinant of a highly structured $(D+M)\times(D+M)$ matrix vanishes. Such a matrix is a special case of what is known as a* **quasi-Toeplitz** *matrix.*

The best general algorithms for evaluating such determinants yields a randomized bit complexity upper bound of $O((D+M)^3\log^\eta(D+M))$, for some absolute constant $\eta>0$ [**EP05**]. *(Gröbner bases, being far more general than what we need, yield a deterministic complexity upper bound no better than $(D+M)^{O(1)}$ bit operations (see, e.g.,* [**Lak91**]*).) More directly, one could also compute the gcd of f and x^M-1, but this still leads to a deterministic bit complexity upper bound no better than $O(DM)$ (see, e.g.,* [**BPR06**, Ch. 8]*). Solving even this special case of* $\texttt{TorsionPoint}_1$ *within $O((D+M)^\varepsilon)$ bit operations for some $\varepsilon\in(0,1)$ is thus still an open problem.* $\diamond$

While the **NP**-hardness of $\texttt{TorsionPoint}_1$ was derived earlier by David A. Plaisted [**Pla84**] in a different context, our complexity upper bounds are new: the best previous upper bounds were **PSPACE** [**Can88**] (unconditionally), $\mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$ [**Roj03**], or **AM** [**Koi96**] (under successively stronger unproved number-theoretic hypotheses, all stronger than APH), following from much more general results. It is also interesting to note that $\texttt{TorsionPoint}_1$ is the same as detecting the vanishing of so-called **cyclic resultants**, which arise in dynamical systems and knot theory [**Hil05**].

Let us now motivate and clarify our use of the term "torsion point" by showing how our results can also be viewed in the context of **Lang's Conjecture** from Diophantine geometry (see, e.g., [**Lan97**, Conj. 6.3, pp. 37–38]).

Notation. *Throughout this paper, we will let $x^a:=x_1^{a_1}\cdots x_n^{a_n}$ and $m\cdot x:=(m_1x_1,\ldots,m_nx_n)$, where it is understood that $a=(a_1,\ldots,a_n)\in\mathbb{Z}^n$, $m=(m_1,\ldots,m_n)\in(\mathbb{C}^*)^n$, and $x=(x_1,\ldots,x_n)\in(\mathbb{C}^*)^n$. Also, given $\bar{d}_1,\ldots,\bar{d}_r\in\mathbb{Z}^n$,* *we let $T(\bar{d}_1,\ldots,\bar{d}_r)$ denote the subgroup of $x\in(\mathbb{C}^*)^n$ satisfying $x^{\bar{d}_1}=\cdots=x^{\bar{d}_r}=1$. We call any point of $(\mathbb{C}^*)^n$ with each coordinate a root of unity a* **torsion point**. *Finally, for any $g_1,\ldots,g_k\in\mathbb{Z}[x_1,\ldots,x_n]$, $Z(g_1,\ldots,g_k)$ denotes the zero set of $g_1,\ldots,g_k$ in $\mathbb{C}^n$.* $\diamond$

The subgroup $T(\bar{d}_1,\ldots,\bar{d}_r)$ is sometimes known in algebraic geometry as a **subtorus**,[1] and the set $m\cdot T(\bar{d}_1,\ldots,\bar{d}_r)$ is usually called a **translated** subtorus. The distribution of torsion points and subtori on algebraic sets happens to be quite special: a given algebraic set will have all its torsion points contained in a subset that is a finite union of subtori, each translated by a torsion point. This follows from a famous result of Laurent [**Lau84**] which was conjectured earlier by Chabauty [**Cha38**]. Explicit bounds on how many torsion points can lie in an algebraic set have been given by Ruppert in certain cases [**Rup93**], and Bombieri and Zannieri in far greater generality [**BZ95**].

Given these deep results, one may suspect that $\texttt{FEAS}_{\mathbb{C}}(\mathcal{F})$ can be sped up when the underlying family $\mathcal{F}$ is restricted to problems involving torsions points. Our two main theorems show that this is indeed the case. In particular, Theorem 1.3 below complements Theorem 1.1 by examining when an algebraic set contains an entire subgroup worth of torsion points, as opposed to a single torsion point. Please note that Theorem 1.3 does **not** depend on any unproved hypotheses.

Theorem 1.3. *Following the notation above, for any $\ell_1,\ldots,\ell_k\in\mathbb{N}$, $\bar{d}_1,\ldots,\bar{d}_r\in\mathbb{Z}^n$ and $f_{i,j}\in\mathbb{Z}[x_1,\ldots,x_n]$ with (i,j) ranging over $\bigcup_{i=1}^k\{(i,1),\ldots,(i,\ell_i)\}$, let* `HasTorus` *denote the problem of deciding whether*

$$T(\bar{d}_1,\ldots,\bar{d}_r)\overset{?}{\subseteq}Z\left(\prod_{j=1}^{\ell_1}f_{1,j},\ldots,\prod_{j=1}^{\ell_k}f_{k,j}\right).$$

Then, measuring the underlying input size **instead** *as*

$$\left(\sum_{i=1}^{r}\text{size}\left(\bar{d}_i\right)\right)+\sum_{i=1}^{k}\sum_{j=1}^{\ell_i}\text{size}(f_{i,j}),$$

we have:

1. `HasTorus` $\in$ **coNP**, *and the restriction of* `HasTorus` *to $n=1$ is already* **coNP**-*hard.*
2. *For* **fixed** *n, $\ell_1,\ldots,\ell_k$, and $\bar{d}_1,\ldots,\bar{d}_r$, we have* `HasTorus` $\in$ **P**.

In particular, `HasTorus` $\notin\mathbf{P}\iff\mathbf{P}\neq\mathbf{NP}$.

Assertions (1) and (2) of Theorem 1.3, in the special case $n=1$, were derived earlier respectively in [**Pla84**] and Theorem 2 of the first ArXiV version of [**BRS07**], but with no reference to tori. Note in particular that our first notion of size for $\prod_{j=1}^{\ell}g_j$ can be exponential in $\sum_{j=1}^{\ell}\text{size}(g_j)$ (e.g., take $g_j:=x_j-1$ for all j), so Theorem 1.3 uses a much more compact notion of input size than Theorem 1.1.

Theorems 1.1 and 1.3 can thus be viewed as first steps toward an algorithmic counterpart to Laurent's Theorem. In particular, having derived nearly tight lower and upper complexity bounds, our results allow us to efficiently detect the presence of subtori. Determining the actual **exceptional locus** — i.e., the precise finite union of translated subtori containing all the torsion points in a given algebraic set — remains an open problem.

Laurent's Theorem has since been extended to algebraic groups more general than $(\mathbb{C}^*)^n$ — semi-Abelian varieties — by McQuillan [**McQ95**], thus solving the

[1] The subtori we consider here need **not** be connected.

aforementioned Lang Conjecture [**Lan97**, Conj. 6.3, pg. 37–38]. For instance, a very special case of McQuillan's more general result is the Faltings-Mordell Theorem. A very special case of the latter result is the fact that an algebraic curve of genus ≥ 2, say, defined as the zero set of a bivariate polynomial with rational coefficients, has at most finitely many rational points.

The existence of algorithmic counterparts to these more general results is thus a tantalizing possibility. An implementable algorithm for finding torsion points on Jacobians of algebraic curves of genus ≥ 2 has already been detailed by Bjorn Poonen [**Poo01**], and the complexity appears (but has not yet been proved) to be polynomial-time for **fixed** genus [**Poo05**]. Such a complexity bound, if proved for the sparse encoding, would form an intriguing analogue to the polynomiality of `TorsionPoint` for **fixed** n and $\bar{d}_1, \ldots, \bar{d}_n$.

In closing this introduction, let us point out that our improved complexity bounds appear to hinge on the highly refined structure of the Galois groups underlying our equations: cyclic. In particular, whereas complex feasibility for an input system F is (conjecturally) solvable by checking the density of primes p for which the mod p reduction of F has a root mod p [**Koi96**], our algorithms instead use a **single** well-chosen p. It is therefore appropriate to formulate the following conjecture, based on an observation of Rachel Pries [**Pri06**]:

Conjecture. *Suppose $\mathcal{F}$ is the family of polynomial systems F such that $Z(F)$ is finite and the Galois group of F over $\mathbb{Q}$ is dihedral or bicyclic. Then the restriction of* $\texttt{FEAS}_{\mathbb{C}}$ *to $\mathcal{F}$ lies in* $\mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$ *unconditionally.*

While the algorithm underlying the general case of Theorem 1.1 is simpler than that of Theorem 1.3, the key ideas flow more clearly if we begin with the latter theorem. So we review some key ideas in one variable in Section 2, and then prove Theorem 1.3 in Section 3 below. We finally prove Theorem 1.1 across Sections 4 and 5, and briefly discuss some limits to possible improvements in Section 6.

1.1. Comparison to Related Results. As mentioned before, our main results improve upon Koiran's earlier algorithms for $\texttt{FEAS}_{\mathbb{C}}$ [**Koi96**] by relaxing, or removing entirely, his assumption of GRH for certain input families. Our success in the setting of torsion points and subtori can hopefully be extended to situations where the underlying Galois groups are more complicated, and membership in the polynomial hierarchy was possible only under stronger assumptions [**Koi96, Roj03**]. We also point out that the work of David Alan Plaisted [**Pla84**] — which focussed on polynomials in one variable — was a central inspiration behind this paper. Our results extend [**Pla84**] to multivariate polynomials and subtori, and suggest the broader context of computational arithmetic geometry [**Roj01**].

One should also remember earlier work of Grigoriev, Karpinski, and Odlyzko [**GKO96**], where it was shown that one can decide if one sparse univariate polynomial divides another, within **coNP**, assuming GRH. Our Theorem 1.3 can be viewed as an unconditional extension of their result to certain multivariate binomial ideals. Needless to say, the results of [**Koi96, Roj03**] contain those of [**GKO96**] as special cases, but the more general results still depend on unproved number-theoretic hypotheses.

Finally, we point out that as this paper was being completed, the author found the paper [**FS04**] during a `MathSciNet` search. In this paper, the authors present a **polynomial-time** algorithm (found by their referee [**FS04**, Thm. 3 and Algor.

A, pp. 959–962; Acknowledgements]) for deciding whether a sparse univariate polynomial is divisible by the $d^{\underline{\text{th}}}$ cyclotomic polynomial for an input integer d **whose factorization is known**. (David A. Plaisted claimed such a result 20 years before [**FS04**], but without a proof [**Pla84**, Top of page 132].) Corollary 1 of [**FS04**] then says that for a **fixed number of monomial terms**, f being divisible by a cyclotomic polynomial implies that f is divisible by the $d^{\underline{\text{th}}}$ cyclotomic polynomial for some d with prime factors bounded above by a constant. Such integers can be factored in polynomial time, and they thus obtain a polynomial time algorithm to decide whether a sparse univariate polynomial f is divisible by **some** cyclotomic polynomial, **when the number of monomial terms of f is fixed** [**FS04**, Thm. 1, pg. 957]. An analogous speed-up for the restriction of $\texttt{TorsionPoint}_1$ to a fixed number of monomial terms appears to remain unknown.

The techniques of [**FS04**] are quite similar to those of [**Pla84**], with two exceptions: (1) [**FS04**] makes no use of certificates in finite fields and (2) [**FS04**] makes clever use of a result of Conway and Jones [**CJ76**] stating in essence that polynomials vanishing at a primitive $d^{\underline{\text{th}}}$ root of unity (with no proper subsum vanishing) can not be "too sparse" as a function of d.

Our techniques complement the results of [**FS04**] by showing that their main problem lies in the polynomial hierarchy unconditionally, even when the number of monomial terms varies and the factorization of d is unknown. This follows directly from our proof of Theorem 1.3, which also extends their context to subtori in higher dimensions.

2. Roots of Unity, Primes, and Illustrative Examples

Definition 2.1. *For any ring R we will let R^* denote the group of multiplicatively invertible elements of R. Also, a* **primitive** *$M^{\underline{th}}$ root of unity is a complex number $\omega \in \mathbb{C}$ such that $\omega^M = 1$ and $[\omega^{M'} = 1 \implies M|M']$. The $M^{\underline{th}}$* **cyclotomic polynomial**, *$\Phi_M \in \mathbb{Z}[x_1]$, is then the minimal polynomial for the primitive $M^{\underline{th}}$ roots of unity.* $\diamond$

Example 2.2. *Specializing Example 1.2 from the Introduction, note that the following assertions are equivalent: (1) f vanishes at an $M^{\underline{th}}$ root of unity, (2) f vanishes at a* **primitive** *$d^{\underline{th}}$ root of unity for some $d|M$, (3) $\Phi_d(x_1)|f(x_1)$ for some $d|M$. For the sake of illustration, let us assume $91|M$ and take $d = 91$. Since $x_1^M - 1 = \prod_{d|M} \Phi_d(x_1)$ for all M (see, e.g.,* [**BS96**, Ch. 6]*), it is then easy to see that f vanishes at a primitive $91^{\underline{st}}$ root of unity $\Longleftrightarrow (x_1^{91}-1)|f(x_1)(x_1^{13}-1)(x_1^7-1)$. The latter condition is in turn equivalent to the truth of*

$$(\star) \qquad \left(x_1^{91c} - 1\right) | f\left(x_1^c\right)\left(x_1^{13c} - 1\right)\left(x_1^{7c} - 1\right)$$

for all $c \in \mathbb{N}$. $\diamond$

Our main algorithmic tricks — when specialized to the example above — are (a) reducing the last check over all $c \in \mathbb{N}$ to a single well chosen c and (b) working over a finite field instead of $\mathbb{Z}[x_1]$. In particular, assuming $91c+1$ is prime, it follows easily from Fermat's Little Theorem that $(\star) \implies f(x_1^c)(x_1^{13c}-1)(x_1^{7c}-1) \equiv 0 \ \mathbf{mod}\ 91c+1$ for all $x_1 \in (\mathbb{Z}/(91c+1)\mathbb{Z})^*$. The multivariate lemma below will later help us derive that the **converse** holds as well, provided c is large enough.

Lemma 2.3. *For any polynomials $g, g_1, \ldots, g_k \in \mathbb{Z}[x_1, \ldots, x_n]$ (expressed as sums of monomial terms), let $\|g\|_1$ denote the sum of the absolute values of the* *coefficients of g, and let $d_i := \deg_{x_i} g$ for all i. Then $\left\|\prod_{j=1}^k g_j\right\|_1 \leq \prod_{j=1}^k \|g_j\|_1$. Also, if q is a prime satisfying $q > \|g\|_1, 1+\max_i\{d_i\}$; and $g(x) \equiv 0$ **mod** q for all $x \in ((\mathbb{Z}/q\mathbb{Z})^*)^n$, then g is identically 0.*

Remark 2.4. *One should recall* **Schwartz's Lemma** [**Sch80**]*, which asserts that for any field K, and any finite subset $S \subseteq K$, a polynomial $g \in K[x_1,\ldots,x_n]$ that is not identically zero vanishes at $\leq (d_1+\cdots+d_n)\#S^{n-1}$ points of S^n. Applying this result would, however, yield a weaker version of the second part of our lemma by requiring a larger q ($q > \sum_i d_i$). Nevertheless, the proof below is quite reminiscent of the proof of Schwartz's Lemma.* $\diamond$

Proof of Lemma 2.3: Writing $g_j(x) = \sum\limits_{a\in A_j} c_{j,a}x^a$ for all j, observe that

$$\left\|\prod_{j=1}^k g_j\right\|_1 = \left\|\prod_{j=1}^k \sum_{a\in A_j} c_{j,a}x^a\right\|_1 = \sum_{\substack{a=a_1+\cdots+a_k\\ a_j\in A_j \text{ for all } j}} \left| \sum_{\substack{(a'_1,\ldots,a'_k)\in(A_1,\ldots,A_k)\\ a'_1+\cdots+a'_k=a}} \prod_{j=1}^k c_{j,a'_j}\right|$$

$$\leq \sum_{\substack{a=a_1+\cdots+a_k\\ a_j\in A_j \text{ for all } j}} \sum_{\substack{(a'_1,\ldots,a'_k)\in(A_1,\ldots,A_k)\\ a'_1+\cdots+a'_k=a}} \prod_{j=1}^k |c_{j,a'_j}| \leq \prod_{j=1}^k \sum_{a_j\in A_j} |c_{j,a_j}| = \prod_{j=1}^k \|g_j\|_1.$$

So the first portion is proved.

We now proceed by induction on n: If $n=1$ then $g(x_1) \equiv 0$ **mod** q for all $x_1 \in (\mathbb{Z}/q\mathbb{Z})^* \implies c_0 \equiv \cdots \equiv c_{d_1} \equiv 0$ **mod** q, since $q-1 > d_1$ and a (not identically zero) polynomial of degree $\leq d_1$ can have at most d_1 roots in $(\mathbb{Z}/q\mathbb{Z})^*$. Since $q > \|h\|_1 \geq \max_i |c_i|$, we thus have $c_0 = \cdots = c_{d_1} = 0$, and our base case is complete.

To conclude, assume that the second portion of our lemma holds for some fixed $n \geq 1$. Let us then temporarily consider g as a polynomial in x_{n+1} with coefficients in $\mathbb{Z}[x_1,\ldots,x_n]$. Let $c_i(x_1,\ldots,x_n)$ denote the coefficient of x^i_{n+1}. Fixing any values for $x_1,\ldots,x_n$, observe that just as in the last paragraph, g can vanish at no more than d_{n+1} values of $x_{n+1} \in (\mathbb{Z}/q\mathbb{Z})^*$. Since $q-1 > d_{n+1}$ we then obtain $c_0(x_1,\ldots,x_n) \equiv \cdots \equiv c_{d_{n+1}}(x_1,\ldots,x_n) \equiv 0$ **mod** q for all $x_1,\ldots,x_n \in (\mathbb{Z}/q\mathbb{Z})^*$. Since $\|c_i(x_1,\ldots,x_n)\|_1 \leq \|h\|_1$ for all i, and since the c_i have exponents no larger than $q-2$, our induction hypothesis then implies that $c_0,\ldots,c_{d_{n+1}}$ are identically 0, and thus g is indeed identically 0. ■

That we can pick a **small** c with $cM+1$ prime is guaranteed by a classic theorem of Linnik.

Linnik's Theorem. *The least prime of the form $cM+b$, where M and b are relatively prime integers and $1 \leq b < M$, does not exceed M^{C_0} for some absolute constant C_0.* ■

The best current unconditional estimate for C_0 is $C_0 \leq 5.5$, assuming M is sufficiently large [**HB92**]. It is also known that the truth of GRH implies that we can take $C_0 = 2+\varepsilon$ for any $\varepsilon > 0$, but of course valid only for $M > M_0$, with M_0 an increasing function of $\frac{1}{\varepsilon}$ [**BS96**, Thm. 8.5.8, pg. 223].

Example 2.5 (A Number-Theoretic Speed-Up). *Let us consider*

$$\begin{aligned} f(x_1) \quad := \quad & x_1^{255255} - 5x_1^{249255} - 3x_1^{248928} + 4x_1^{234655} - 5x_1^{221135} + 2x_1^{213883} - x_1^{210952} + 4x_1^{200774} \\ & +4x_1^{199666} - 5x_1^{191411} + 5x_1^{187436} + 2x_1^{186678} + 3x_1^{181717} - 4x_1^{181453} + 5x_1^{180273} + 3x_1^{176054} \\ & +3x_1^{171282} - 4x_1^{170662} + 3x_1^{168177} + x_1^{164270} + 5x_1^{157315} + 2x_1^{154380} + 5x_1^{147177} - 2x_1^{144498} \\ & -4x_1^{142969} - 2x_1^{139399} + 3x_1^{127018} + 3x_1^{103857} - 4x_1^{101698} + x_1^{97641} + 2x_1^{91638} - 5x_1^{88391} \\ & -5x_1^{88198} - 4x_1^{86818} + 5x_1^{85759} + 5x_1^{73803} - x_1^{64076} - 3x_1^{60689} - 2x_1^{50793} - 5x_1^{24214} + 4x_1^{22380} \\ & -2x_1^{12176} - 5x_1^{682} - 2, \end{aligned}$$

which has degree 255255 *and exactly* 46 *monomial terms, and suppose we'd like to verify whether f vanishes at some $510510^{\underline{th}}$ root of unity. To illustrate our approach via cyclotomic polynomials, let us first see if f vanishes at a* **primitive** *$91^{\underline{st}}$ root of unity. As observed earlier, when $q := 91c+1$ is prime, we have that $(x_1^{91c}-1) | f(x_1^c)(x^{13c}-1)(x^{7c}-1) \implies f(t^c)(t^{13c}-1)(t^{7c}-1) \equiv 0$* **mod** *$q$ for all $t \in (\mathbb{Z}/q\mathbb{Z})^*$. So Condition ($\star$) implies a certain congruence holds. However, the reduction goes the other way as well: Lemma 2.3 (applied to the* **mod $\boldsymbol{t^{91}-1}$ reduction** *of $f(t)(t^{13}-1)(t^7-1)$) tells us that the* **converse** *to the preceding implication holds, provided q is prime, $q > \|f\|_1 \|x_1^{13}-1\|_1 \|x_1^7-1\|_1 = 568$, and $q > 255256$.*

In particular, $2842 \cdot 91 + 1 = 258623$ is prime. So to check whether f vanishes at a primitive $91^{\underline{st}}$ root of unity, we need only check whether

$$f\big(t^{2842}\big)\left(t^{2842\cdot 13}-1\right)\left(t^{2842\cdot 7}-1\right) \overset{?}{\equiv} 0 \ \mathbf{mod}\ 258623 \textit{ for all } t \in (\mathbb{Z}/258623\mathbb{Z})^*.$$

Since $t = 3$ yields 76177 *for the product polynomial above, we thus have certification that f does* **not** *vanish at any primitive $91^{\underline{st}}$ root of unity. Similar calculations for* **small** *choices of c and t then suffice to show that f does not vanish at any primitive $d^{\underline{th}}$ root of unity for any other $d|510510$ either. (Excluding the easy case $d=1$ and the case $d=91$ we just did, there are exactly* 126 *other such cases.) Thus, we can at last certify that f does not vanish at* **any** *$510510^{\underline{th}}$ root of unity.* $\diamond$

It is easily checked that the number of bit operations for the calculations of Example 2.5 (including the work for the additional 126 cases of $d|510510$) lies in the lower hundreds of thousands. (This is via standard mod n arithmetic (see, e.g., [**BS96**, Ch. 5]), with **no** use of FFT multiplication.) More concretely, the finite field certificate check above took but a fraction of a second.[2] On the other hand, computing the gcd of $x^{510510}-1$ and the f above took 37 minutes and 38.9 seconds.[2] We analyze the underlying asymptotic complexity in greater depth in the next section, where we also formalize our algorithm for `HasTorus`.

3. Complexity Issues and Proving Theorem 1.3: Detecting Subtori Unconditionally

Let us recall the following informal descriptions of some famous complexity classes. A completely rigourous and detailed description of the classes below can be found in the excellent reference [**Pap95**]. Our underlying computational model is the classical Turing model. For concreteness, it is not unrealistic to simply imagine that we are working with a laptop computer, equipped with infinite memory, flawless hardware, and a flawless operating system: classical theorems from complexity theory allow one to define the complexity classes below in a machine-independent

[2] Using the computer algebra system `Maple 9.5`, on `diana`, the author's 4Gb dual-Athlon 2 Ghz Fedora Core 4 Linux system.

manner. (We omit these more formal definitions for brevity). In particular, we can identify "time" or "work" with how long our laptop computer takes to solve a given problem, and "input size" can simply be identified with the number of bytes in some corresponding input file.

P The family of decision problems which can be done within time polynomial in the input size.[3]

BPP The family of decision problems admitting randomized algorithms that terminate in polynomial-time to give an answer which is correct with probability at least[4] $\frac{2}{3}$.

NP The family of decision problems where a ‘‘Yes’’ answer can be **verified** within time polynomial in the input size.

coNP The family of decision problems where a ‘‘No’’ answer can be **verified** within time polynomial in the input size.

AM The family of decision problems solvable by a **BPP** algorithm which has been augmented with exactly **one** use of an oracle in **NP**.

$\mathbf{NP}^{\mathbf{NP}}$ The family of decision problems where a ‘‘Yes’’ answer can be certified by using an **NP**-oracle a number of times polynomial in the input size.

$\mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$ The family of decision problems solvable within time polynomial in the input size, with as many calls to an $\mathbf{NP}^{\mathbf{NP}}$ oracle as allowed by the time bound.

PSPACE The family of decision problems solvable within time polynomial in the input size, provided a number of processors exponential in the input size is allowed.

EXPTIME The family of decision problems solvable within time exponential in the input size.

The inclusions

$$\mathbf{P} \subseteq \mathbf{BPP} \cup \mathbf{NP} \subseteq \mathbf{AM} \subseteq \mathbf{coNP}^{\mathbf{NP}} \subseteq \mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$$

and

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{NP}^{\mathbf{NP}} \subseteq \mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME},$$

are fundamental in complexity theory [**Pap95, BM88**], and the properness of every explicitly stated inclusion above turns out to be a major open problem (as of late 2007). For instance, while we know that $\mathbf{P} \subsetneqq \mathbf{EXPTIME}$, the inclusion $\mathbf{P} \subseteq \mathbf{PSPACE}$ is not even known to be proper. The first 6 complexity classes in the list above lie in a family known as the **polynomial hierarchy**. It is known that $\mathbf{P} = \mathbf{NP}$ implies that the polynomial hierarchy **collapses**, which in particular yields the equalities $\mathbf{P} = \mathbf{NP} = \mathbf{coNP} = \mathbf{AM} = \mathbf{NP}^{\mathbf{NP}} = \mathbf{P}^{\mathbf{NP}^{\mathbf{NP}}}$ [**Pap95**, Thm. 17.9]. This standard fact will be used later.

The structure of our main algorithms depends on a useful number-theoretic lemma stated below. In what follows, e_i denotes the $i^{\underline{\text{th}}}$ standard basis vector of whatever finite-dimensional module we are working in.

Definition 3.1. *For any $g \in \mathbb{Z}[x_1, \ldots, x_n]$, let $\bar{g} \in \mathbb{Z}[x_1, \ldots, x_n]$ denote the polynomial obtained by reducing all exponent vectors in the monomial term expansion of g modulo the additive subgroup $\langle d_1 e_1 \ldots, d_r e_r \rangle$ of $\mathbb{Z}^n$ and collecting terms.* $\diamond$

[3] Note that the underlying polynomial depends only on the problem in question (e.g., matrix inversion, shortest path finding, primality detection) and not the particular instance of the problem.

[4] It is easily shown that we can replace $\frac{2}{3}$ by any constant strictly greater than $\frac{1}{2}$ and still obtain the same family of problems.

Note that computing $\bar{g}$ is nothing more than repeatedly applying the substitution $x_i^{d_i}=1$ (for all monomial terms and $i\in\{1,\ldots,r\}$), and simplifying, until one obtains a polynomial with degree $<d_i$ with respect to x_i for all $i\in\{1,\ldots,r\}$. Note also that any coefficient of $\bar{g}$ is a sum of coefficients of g.

Proposition 3.2. *For any $g,g_1,\ldots,g_\ell\in\mathbb{Z}[x_1,\ldots,x_n]$, let m_j denote the number of monomial terms of g_j for all j. Then $\|\bar{g}\|_1\leq\|g\|_1$, and the monomial term expansion of $\overline{\prod_{j=1}^\ell g_j}$ can be computed within*

$$O\left(\min\left\{\prod_{j=1}^{\ell} m_j, \prod_{i=1}^{r} d_i\right\}\left(\sum_{j=1}^{\ell}\text{size}(g_j)+\sum_{i=1}^{r}\log d_i\right)^2\right)$$

bit operations.

Proof: The first portion follows directly from the definition of $\|\cdot\|_1$ and $\bar{g}$.

To prove the second portion, note that computing $\bar{g}_j$ consists simply of reducing the coordinates of the exponent vectors modulo integers of size no larger than $\max_i\{\log d_i\}$, and then summing up coefficients of monomial terms. So via basic fast finite field arithmetic (e.g., [**BS96**, Table 3.1, Pg. 43]), this can be done within $O\left(\sum_{i=r}^{r}\max\{\text{size}(g_j),\log(d_i)\}\log\max\{\text{size}(g_j),\log(d_i)\}\right)$ bit operations.

Next, note that to compute $\overline{\prod_{j=1}^\ell g_j}$, we can use the recurrence $G_1:=\bar{g}_1$, $G_{j+1}=\overline{G_j\bar{g}_{j+1}}$, and stop at G_ℓ. Defining κ_j to be the maximum bit-length of any coefficient of $\bar{g}_j$, the number of bit operations to compute G_2 is then easily seen to be $O^*(\min\{m_1m_2,\prod_{i=1}^r d_i\}(\kappa_1+\kappa_2+\sum_{i=1}^r\log d_i))$. (The $O^*(\cdot)$ notation indicates that additional factors polynomial in $\log\kappa_j$ and $\log\log d_i$ are omitted.) This bound is obtained by first computing $\bar{g}_1\bar{g}_2$ by simply multiplying all monomials of $\bar{g}_1$ with all monomials of $\bar{g}_1$ (using fast arithmetic along the way), collecting terms, and then reducing the exponents as in the definition of $\overline{(\cdot)}$. Continuing inductively, our complexity bound follows directly, keeping in mind that $\left\|\overline{\prod_{j=1}^\ell g_j}\right\|_1\leq\left\|\prod_{j=1}^\ell \bar{g}_j\right\|_1\leq\prod_{j=1}^\ell\|\bar{g}_j\|_1\leq\prod_{j=1}^\ell\|g_j\|_1$. ■

Lemma 3.3. *Following the notation above, suppose $g\in\mathbb{Z}[x_1,\ldots,x_n]$, $d_1,\ldots,d_r\in\mathbb{N}$, $D:=2+\max\{\max_{i\in\{1,\ldots,r\}}\{d_i\},\max_{i\in\{r+1,\ldots,n\}}\{\deg_{x_i}g\}\}$, $M:=\left\lceil\frac{\max\{\|g\|_1,D\}}{\text{lcm}_i\{d_i\}}\right\rceil\text{lcm}_i\{d_i\}$, and assume c is a positive integer such that $q:=cM+1$ is prime. Then*

$$T(d_1e_1,\ldots,d_re_r)\subseteq Z(g)\Longleftrightarrow\begin{cases}g\left(t_1^{cM/d_1},\ldots,t_r^{cM/d_r},t_{r+1},\ldots,t_n\right)\equiv 0 \textbf{ mod } q\\ \text{for all } t_1,\ldots,t_n\in(\mathbb{Z}/q\mathbb{Z})^*.\end{cases}$$

Proof: Let J denote the ideal $\langle x_1^{d_1}-1,\ldots,x_r^{d_r}-1\rangle\subset\overline{\mathbb{Q}}[x_1,\ldots,x_n]$. Observe that the primary decomposition of J is clearly $\bigcap_{\zeta_1^{d_1}=\cdots=\zeta_r^{d_r}=1}\langle x_1-\zeta_1,\ldots,x_r-\zeta_r\rangle$, and each ideal in the preceding intersection is prime. J is thus a radical ideal in $\overline{\mathbb{Q}}[x_1,\ldots,x_r]$. Now let $I:=J\cap\mathbb{Q}[x_1,\ldots,x_n]$. Before proving our desired equivalence we will need the fact that the ideal I of $\mathbb{Q}[x_1,\ldots,x_n]$ is radical as well. So let us conclude this necessary digression as follows:

Suppose $f^k\in I$ for some $f\in\mathbb{Q}[x_1,\ldots,x_n]$ and $k>1$. Since J is radical and $J\supseteq I$, we then clearly obtain the existence of $f_1,\ldots,f_r\in\overline{\mathbb{Q}}[x_1,\ldots,x_n]$ with

$$f(x)=(x_1^{d_1}-1)f_1(x)+\cdots+(x_r^{d_r}-1)f_r(x).$$

Letting G denote the Galois group of the coefficients of the f_i over $\mathbb{Q}$, let us define $f'_i := \frac{1}{\#G}\sum_{\sigma\in G}\sigma(f_i)$ for all $i\in\{1,\ldots,r\}$. Observe then that $f(x)$ also equals $\sum_{i=1}^r(x_i^{d_i}-1)f'_i(x)$, and thus lies in I as well by Galois invariance. So I is radical.

Returning to our main proof, we now see that:

(A) $T(d_1e_1,\ldots,d_re_r)\subseteq Z(g) \Longleftrightarrow g\in I$, and

(B) $\{x^a \mid a\in\{0,\ldots,d_1-1\}\times\cdots\times\{0,\ldots,d_r-1\}\}$ is a $\mathbb{Q}$-vector space basis for $\mathbb{Q}[x_1,\ldots,x_r]/I$.

In particular, $T(d_1e_1,\ldots,d_re_r)\subseteq Z(g)$ iff $\bar{g}$ is identically zero. So it suffices to prove that $\bar{g}$ is identically zero $\Longleftrightarrow g\Big(t_1^{cM/d_1},\ldots,t_r^{cM/d_r},t_{r+1},\ldots,t_n\Big) \equiv 0 \textbf{ mod } q$ for all $t_1,\ldots,t_n\in(\mathbb{Z}/q\mathbb{Z})^*$. Let $I_{cM}:=\langle x_1^{cM}-1,\ldots,x_r^{cM}-1\rangle$.

($\Longrightarrow$): By (B), $\bar{g}$ identically zero $\Longrightarrow g\in I$, and thus $g(x_1^{cM/d_1},\ldots,x_r^{cM/d_r},x_{r+1},\ldots,x_n)\in I_{cM}$. Since $q:=cM+1$ is prime, Fermat's Little Theorem implies $t^{cM}-1\equiv 0 \textbf{ mod } q$ for all $t\in\{1,\ldots,cM\}$, so

$$g(t_1^{cM/d_1},\ldots,t_r^{cM/d_r},t_{r+1},\ldots,t_n)\equiv 0 \text{ mod } q, \text{ for all } t_1,\ldots,t_n\in(\mathbb{Z}/q\mathbb{Z})^*.$$

(Remember that we have defined M so that $d_i|M$ for all $i\in\{1,\ldots,r\}$.)

($\Longleftarrow$): By (B), $g-\bar{g}\in I$ for any $g\in\mathbb{Z}[x_1,\ldots,x_n]$. So we must then have

$$g\Big(x_1^{cM/d_1},\ldots,x_r^{cM/d_r},x_{r+1},\ldots,x_n\Big)-\bar{g}\Big(x_1^{cM/d_1},\ldots,x_r^{cM/d_r},x_{r+1},\ldots,x_n\Big)\in I_{cM}.$$

We therefore obtain that $g\left(t^{cM/d_1},\ldots,t_r^{cM/d_r},t_{r+1},\ldots,t_n\right) \textbf{ mod } q$ for all $t_1,\ldots,t_n\in(\mathbb{Z}/q\mathbb{Z})^* \Longrightarrow \bar{g}\Big(t_1^{cM/d_1},\ldots,t_r^{cM/d_r},t_{r+1},\ldots,t_n\Big)\equiv 0 \textbf{ mod } q$ for all $t_1,\ldots,t_n\in(\mathbb{Z}/q\mathbb{Z})^*$, via another application of Fermat's Little Theorem.

Now note that $\|\bar{g}\|_1\leq\|g\|\leq M\leq q$ and

$$\deg_{x_i}\bar{g}\Big(x_1^{cM/d_1},\ldots,x_r^{cM/d_r},x_{r+1},\ldots,x_n\Big)\leq(cM/d_i)(d_i-1)<cM=q-1$$

for all $i\in\{1,\ldots,r\}$. Furthermore, $\deg_{x_i}\bar{g}=\deg_{x_i}g\leq D-2\leq M<q-1$ for all $i\in\{r+1,\ldots,n\}$. So Lemma 2.3 immediately implies that $\bar{g}$ is identically 0. ■

We now state our first main algorithm.

Algorithm 3.4 (For problem `HasTorus`, with simplified subtori, unconditionally).

Input: *Polynomials* $f_{i,j}\in\mathbb{Z}[x_1,\ldots,x_n]$ *with* $(i,j)\in\bigcup_{i=1}^k\{(i,1),\ldots,(i,\ell_i)\}$, *positive integers* $d_1,\ldots,d_r$, *and a suitable value for the constant* C_0 *from Linnik's Theorem.*

Output: *A true declaration of whether*

$$Z\Big(\prod_{j=1}^{\ell_1}f_{1,j},\ldots,\prod_{j=1}^{\ell_k}f_{k,j}\Big)\supseteq Z(x_1^{d_1}-1,\ldots,x_r^{d_r}-1).$$

Description:

(0) *Replace each* $f_{i,j}$ *by* $\bar{f}_{i,j}$ *(following the notation above).*

(1) *Let* $N:=\max_i\left\{\prod_{j=1}^{\ell_i}\|f_{i,j}\|_1\right\}$ *and* $M:=\left\lceil\frac{\max\{N,D\}}{\operatorname{lcm}_i\{d_i\}}\right\rceil\operatorname{lcm}_i\{d_i\}$, *where* D *is* $2+\max\left\{\max_{j\in\{1,\ldots,r\}}\{d_j\},\max_{(i,j)\in\{1,\ldots,k\}\times\{r+1,\ldots,n\}}\left\{\sum_{s=1}^{\ell_i}\deg_{x_j}f_{i,s}\right\}\right\}$.

(2) *Nondeterministically, decide whether there is a* $c\in\mathbb{N}$ *with* $c\leq M^{C_0}$ *and* $q:=cM+1$ *prime, a* $t=(t_1,\ldots,t_n)\in((\mathbb{Z}/q\mathbb{Z})^*)^n$, *and an* $i\in\{1,\ldots,k\}$, *such that*

$$(\heartsuit_i)\qquad \prod_{j=1}^{\ell_i}f_{i,j}\Big(t_1^{cM/d_1},\ldots,t_r^{cM/d_r},t_{r+1},\ldots,t_n\Big)\not\equiv 0 \textbf{ mod } q.$$

(3) *If the desired (c,t,i) from Step 2 exists then stop and output* `''NO''`. *Otherwise, stop and output* `''YES''`. $\diamond$

The adverb "nondeterministically" can be interpreted in two ways: the simplest is to just ignore the word and employ brute-force search. This leads to an algorithm which is dramatically simpler and easier to implement than resultants or Gröbner bases. All of our examples were handled this simple way, and the respective timings were already competitive with the latter techniques (cf. Examples 1.2, 2.2, 2.5, and 5.4).

Alternatively, one can observe that Step 2 is equivalent to deciding the truth of a quantified Boolean sentence of the form $\forall y_1 \cdots \forall y_\nu B(y_1,\ldots,y_\nu)$, with $B(y_1,\ldots,y_\nu)$ computable in time polynomial in the size of our initial input. This is clarified in our proof of Theorem 1.3 below.

Before starting our proof, we will need a lemma on integral matrices to quantify certain monomial changes of variables.

Definition 3.5. *Let $\mathbb{Z}^{m\times n}$ denote the set of $m\times n$ matrices with all entries integral, and let $\mathbb{GL}_m(\mathbb{Z})$ denote the set of all matrices in $\mathbb{Z}^{m\times m}$ with determinant ± 1 (the set of* **unimodular** *matrices). Recall that any $m\times n$ matrix $[u_{ij}]$ with $u_{ij}=0$ for all $i>j$ is called* **upper triangular**. *Then, given any $M\in\mathbb{Z}^{m\times n}$, we call an identity of the form $UM=H$, with $H=[h_{ij}]\in\mathbb{Z}^{n\times n}$ upper triangular and $U\in\mathbb{GL}_m(\mathbb{Z})$, a* **Hermite factorization** *of M. Also, if we have the following conditions in addition:*

(1) *$h_{ij}\geq 0$ for all i,j.*
(2) *for all i, if j is the smallest j' such that $h_{ij'}\neq 0$ then $h_{ij}>h_{i'j}$ for all $i'\leq i$.*

then we call H **<u>the</u> Hermite normal form** *of M.* $\diamond$

A **Smith** factorization is a more refined factorization of the form $UMV=S$ with $U\in\mathbb{GL}_m(\mathbb{Z})$, $V\in\mathbb{GL}_n(\mathbb{Z})$, and S diagonal. In particular, if $S=[s_{i,i}]$ and we require additionally that $s_{i,i}\geq 0$ and $s_{i,i}|s_{i+1,i+1}$ for all $i\in\{1,\ldots,\min\{m,n\}\}$ (setting $s_{\min\{m,n\}+1,\min\{m,n\}+1}:=0$), then such a factorization for M is unique and is called **the** Smith factorization.

Lemma 3.6. [**Ili89, Sto98**] *For any $A=[a_{ij}]\in\mathbb{Z}^{n\times n}$, the Hermite and Smith factorizations of A can be computed within $O(n^4\log^3(n\max_{i,j}|a_{ij}|))$ bit operations. Furthermore, the entries of all matrices in these factorizations have bit size $O(n^3\log^2(2n+\max_{i,j}|a_{ij}|))$.* $\blacksquare$

Proof of Theorem 1.3: Define $X:=Z\left(\prod_{j=1}^{\ell_1} f_{1,j},\ldots,\prod_{j=1}^{\ell_k} f_{k,j}\right)$ and let us first reduce to the special case where $\bar{d}_i=d_ie_i$ for all i: Let M be the matrix whose columns are $\bar{d}_1,\ldots,\bar{d}_r$ and define $x^M:=(x^{\bar{d}_1},\ldots,x^{\bar{d}_r})$. An elementary calculation then reveals that if we have the Smith factorization $UMV=S=:[s_{i,j}]$ (with S having exactly t nonzero entries), then $x^M=1 \iff (z_1^{s_{1,1}},\ldots,z_t^{s_{t,t}})=(1,\ldots,1)$, upon setting $x:=z^U$. Via Lemma 3.6, we see that this change of variables can be found within $\mathbf{P}$ and the increase in our input size is polynomial in $O(\text{size}(\bar{d}_1)+\cdots+\text{size}(\bar{d}_n))$. So let us assume henceforth that $\bar{d}_i=s_ie_i$ (and let $d_i=s_i$) for all $i\in\{1,\ldots,t\}$ and set $r=t$.

The equivalence of `HasTorus` $\not\in\mathbf{P}$ and $\mathbf{P}\neq\mathbf{NP}$ follows immediately from our earlier remarks on the polynomial hierarchy [**Pap95**, Thm. 17.9], assuming we indeed have `HasTorus` $\in$ **coNP**. So let us proceed with proving Assertions (1) and (2).

Assertion (1): The **coNP**-hardness of the $n=1$ restriction of `HasTorus` — stated equivalently as a problem involving sparse polynomial division — is essentially [**Pla84**, Thm. 4.1]. So we need only show that `HasTorus` $\in$ **coNP** for general n and, thanks to our preceding reductions, this can be done by proving that Algorithm 3.4 is correct and runs within **coNP**.

Correctness follows immediately from Lemma 3.3 applied to the polynomials from $(\heartsuit_1), \ldots, (\heartsuit_k)$.

To analyze the complexity of Algorithm 3.4, first note that Steps 0 and 1 can clearly be done in polynomial time and Step 3 takes essentially constant time. So it suffices to focus on the complexity of Step 2.

Let us then observe that for any $t_1, \ldots, t_n \in \mathbb{Z}/q\mathbb{Z}$, we can verify $(\heartsuit_i)$ in polynomial-time: By basic finite field arithmetic (see, e.g., [**BS96**, Ch. 5]), we can clearly decide within **P** whether any $f_{i,j}$ vanishes at a given point in $(\mathbb{Z}/q\mathbb{Z})^n$ using a number of bit operations polynomial in $\text{size}(g) \log q$, and we then simply multiply the appropriate $f_{i,j}$. In total, checking $(\heartsuit_1), \ldots, (\heartsuit_k)$ at any given point in $(\mathbb{Z}/q\mathbb{Z})^n$ requires a number of bit operations at worst k times a polynomial in $\log(q) \left[\left(\sum\limits_{i=1}^{r} \text{size}(d_i)\right) + \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \text{size}(f_{i,j})\right]$.
Now observe that $\text{size}(q) = O(\log M) = O(\log(N) + \log(D) + \sum_{i=1}^{r} \log d_i)$, which is clearly linear in our input size. Note also that the integer N from Algorithm 3.4 (which by definition is no larger than M) is clearly an upper bound on the 1-norms of the polynomials from $(\heartsuit_1)$, …, $(\heartsuit_k)$. So any instance of inequality $(\heartsuit_i)$ can clearly be checked in **P**.

Now note that verifying $q = cM + 1$ is indeed prime can be done in time polynomial in $\log q$ (which is in turn polynomial in our input size): One can either use the succinct primality certificates of Pratt [**Pra75**], or the deterministic polynomial-time primality testing algorithm from [**AKS04**]. So Step 2 is nothing more than verifying the truth of the following quantified sentence:

$$\exists c \exists t_1 \cdots \exists t_n \exists i \left[(cM+1 \text{ prime}) \wedge (c \leq M^{C_0}) \wedge (\heartsuit_i)\right].$$

X contains the subtorus $T(d_1e_1, \ldots, d_re_r)$ iff the preceding sentence is **false**. So via our preceding observations, the truth of the sentence being quantified can be verified in **P**, and our algorithm thus runs in **coNP**.

Assertion (2): Suppose n, $\ell_1, \ldots, \ell_k$, $d_1, \ldots, d_n$ are fixed. Then, by Proposition 3.2 (with ℓ constant), we can decide `HasTorus` in **P** simply by reducing the exponents modulo suitable integers and doing a brute-force check of the congruence condition given by Lemma 3.3. ■

Example 3.7. *While it is tempting to propose a variant of Algorithm 3.4 to detect* **translated** *subtori, here is an example showing that at least one naive extension breaks down quickly: Suppose $q = kd + 1$ is prime (we can take $k \geq 2$ and arbitrary large by Linnik's Theorem), $g(x) := \gamma x^{dq} - 1$ with $\gamma \equiv 2^d$ **mod** q and $\gamma \in \{1, \ldots, q-1\}$; and we want to see if g vanishes at* **half** *of every $d^{\underline{th}}$ root of unity. Since this happens iff $g(x/2)$ vanishes at every $d^{\underline{th}}$ root of unity, we could try to mimic Algorithm 3.4 by checking whether $g(t^{(q-1)/d}/2) \equiv 0$ **mod** q for all $t \in (\mathbb{Z}/q\mathbb{Z})^*$. This is indeed so, since $g(t^{(q-1)/d}/2) = \gamma \cdot \frac{t^{(q-1)q}}{2^{dq}} - 1 \equiv 2^d \cdot \frac{1}{2^d} - 1 \equiv 0$ **mod** q. However, $g(\zeta/2) = \frac{\gamma}{2^{(q-1)d}} - 1 \neq 0$ for ζ any $d^{\underline{th}}$ root of unity.* ◇

4. From Subtori to Torsion Points: Theorem 1.1 in One Variable, Unconditionally

With some modifications, Algorithm 3.4 — which we used to detect subtori — can be used to efficiently find torsion points in the univariate case.

Algorithm 4.1 (For $\texttt{TorsionPoint}_1$, unconditionally).
Input: *Polynomials* $f_1,\ldots,f_k \in \mathbb{Z}[x_1]$ *and a positive integer* d.
Output: *A true declaration of whether* $Z(f_1,\ldots,f_k)$ *contains a point* ζ *with* $\zeta^d = 1$.
Description:

```
(1) Using Algorithm 3.4, nondeterministically decide whether there is a $\delta|d$
    with $Z(f_1 g_\delta,\ldots,f_k g_\delta) \supseteq Z(x_1^\delta - 1)$ where
        $g_\delta(x_1) := \prod_{p \text{ a prime dividing } \delta} \left(x_1^{\delta/p} - 1\right)$.
(2) If the desired $\delta$ from Step 1 exists then stop and output ``YES''.
    Otherwise, stop and output ``NO''. ◇
```

Just as in our last algorithm, the adverb "nondeterministically" can be interpreted in two ways: first, one can simply employ brute-force search, and this strategy is dramatically simpler and easier to implement than resultants or Gröbner bases. All of our examples were handled in this simple way, and the respective timings were already competitive with the latter techniques (cf. Examples 1.2, 2.2, 2.5, and 5.4).

Alternatively, one can observe that Step 1 is equivalent to deciding the truth of a quantified Boolean sentence of the form $\exists y_1 \cdots \exists y_{\nu'} \forall y_{\nu'+1} \cdots \forall y_\nu B(y_1,\ldots,y_\nu)$, with $B(y_1,\ldots,y_\nu)$ computable in time polynomial in the size of our initial input. This type of sentence forms one of the definitions of the complexity class $\mathbf{NP}^{\mathbf{NP}}$.

Proof of Assertion (2) of Theorem 1.1: The **NP**-hardness of $\texttt{TorsionPoint}_1$ is already implicit in the proof of [**Pla84**, Thm. 5.1], so we need only show that $\texttt{TorsionPoint}_1 \in \mathbf{NP}^{\mathbf{NP}}$. To do the latter, we will prove the correctness of Algorithm 4.1 and that it indeed runs within $\mathbf{NP}^{\mathbf{NP}}$.

The correctness of Algorithm 4.1 follows immediately from Step 1 and the correctness of Algorithm 3.4. In particular, it is clear that f_i vanishes at a primitive $\delta^{\underline{\text{th}}}$ root of unity (indeed, at **all** primitive $\delta^{\underline{\text{th}}}$ roots of unity) iff $(x_1^\delta - 1) | f(x_1) g_\delta(x_1)$.

Recalling that we've already proved that Algorithm 3.4 runs in **coNP** in the last section, Step 1 thus consists of a single existential quantifier calling a **coNP** algorithm. In particular, verifying that a putative δ satisfies $\delta|d$ can clearly be done in **P**, and thus Algorithm 4.1 runs in $\mathbf{NP}^{\mathbf{NP}}$. ■

Remark 4.2. *One can show that the number of possible* δ *dividing* d *in Step (1) of Algorithm 4.1 is* $O(d^\varepsilon)$ *(for any* $\varepsilon > 0$*),* $O((\log d)^{\log(2)+\varepsilon})$ *for a fraction of integers approaching* 1 *as* $d \longrightarrow \infty$ *(for any* $\varepsilon > 0$*), and* $O(\log d)$ *on average. This follows easily from earlier estimates on the number of divisors of an integer (see, e.g.,* [**HR17, NR83, DN94**] *and the references therein). Practically speaking, this means that the main complexity bottleneck in Algorithm 4.1 is the efficient detection of cyclotomic factors.* ◇

Before moving to the higher-dimensional case of $\texttt{TorsionPoint}$, let us point out that the product trick underlying Algorithm 4.1 does not naively extend to $n > 1$.

Example 4.3. *Since $1+\omega_3+\omega_3^2=0$ for any primitive third root of unity ω_3, we see that $1+x+y$ vanishes at a point with coordinates third roots of unity. Can we derive a (polynomial-time certifiable) criterion to detect this, in the spirit of Lemma 2.3 or Step 1 of Algorithm 4.1?*

As an initial attempt, one could first consider the product

$$(1+x+y)(x-1)(y-1)$$

(based on mimicking the use of f_ig_δ in Algorithm 4.1) and see if it lies in the ideal $\langle x^3-1, y^3-1\rangle$. The preceding product, unfortunately, fails this criterion.

On the other hand, the larger product

$$(1+x+y)(1+x+y^2)(x-1)(y-1)$$

does *lie in the ideal $\langle x^3-1, y^3-1\rangle$. However, the most obvious extension of the latter product results in a certificate which can have exponentially many factors in general.* ⋄

While the latter idea does not obviously yield an efficient higher-dimensional extension of Algorithm 4.1, it does enable one to prove the correctness of a **different** (and efficient) higher-dimensional extension of Algorithm 4.1. This we now detail.

5. Completing the Proof of Theorem 1.1

Let us first state an important quantitative result, which follows directly from the effective arithmetic Nullstellensatz of Krick, Pardo, and Sombra [**KPS01**].

Theorem 5.1. *Suppose $f_1,\ldots,f_k\in Z[x_1,\ldots,x_n]$, $d_1,\ldots,d_n$ are positive integers, $F:=(f_1,\ldots,f_k)$, $E:=\max\{\max_i \deg f_i, \max_i d_i\}$, and $\sigma(F)$ is one plus the maximum of the absolute value of the log of any coefficient of any f_i. Then $F(x)=x_1^{d_1}-1=\cdots=x_n^{d_n}-1=0$ has* **no** *complex roots iff there are polynomials $g_1,\ldots,g_k,h_1,\ldots,h_n\in\mathbb{Z}[x_1,\ldots,x_n]$, and a positive integer α, with*

($\star\star$) $$g_1(x)f_1(x)+\cdots+g_k(x)f_k(x)+h_1(x)(x_1^{d_1}-1)+h_n(x)(x_n^{d_n}-1)=\alpha$$

identically, and

(1) $\deg g_i, \deg h_i\leq 2n^2E^{n+1}$

(2) $\log\alpha\leq 2(n+1)^3E^{n+1}(\sigma(F)+\log(k+n)+14(n+1)E\log(E+1))$ ■

Since α has no more than $1+\log\alpha$ prime factors, it is clear that the identity ($\star\star$) persists — with a **nonzero** right-hand side — even after reduction modulo a prime, for all but finitely many primes. This in turn easily implies that lacking torsion points (for fixed degree) is a property that persists as one passes from $\mathbb{C}$ to most finite fields, and the number of exceptions is no more than one plus the right-hand side of Inequality (2) above. The following lemma shows how **possessing** torsion points persists as one passes from $\mathbb{C}$ to certain special finite fields.

Lemma 5.2. *Following the notation of Theorem 5.1, suppose $f_1(x)=\cdots=f_k(x)=x_1^{d_1}-1=\cdots=x_n^{d_n}-1=0$ has a complex root. Then the mod q reduction of the preceding system has a root in $(\mathbb{Z}/q\mathbb{Z})^n$ for any q with $q\equiv 1$* **mod** *$\mathrm{lcm}\{d_1,\ldots,d_n\}$ and q prime.*

Proof: Letting $F=(f_1,\ldots,f_k)$, note that $Z(F)$ has a torsion point of the specified type iff $Z(F)$ contains a point $\zeta=(\zeta_1,\ldots,\zeta_n)$ with ζ_i a primitive δ_i^{th} root of unity, for some positive integers $\delta_1,\ldots,\delta_n$ with $\delta_i|d_i$ for all i. Note then that the polynomial

$$h_i(x_1,\ldots,x_n):=\prod_{\substack{(j_2,\ldots,j_n)\\ j_s \text{ coprime to } \delta_s \forall s\in\{2,\ldots,n\}}} f_i(x_1,x_2^{j_2},\ldots,x_n^{j_n})$$

must satisfy $Z(h_i(x_1,\ldots,x_n)g_{\delta_1}(x_1)\cdots g_{\delta_n}(x_n)) \supseteq T(\delta_1 e_1,\ldots,\delta_n e_n)$ for all i, where g_δ is the polynomial defined in Step 1 of Algorithm 4.1.

Now suppose $q := c\cdot\mathrm{lcm}\{d_1,\ldots,d_n\}+1$ is prime. Then, via the ($\Longrightarrow$) portion of Lemma 3.3 (which, visible from its proof, does **not** require any assumptions on the coefficient size), we must have $h_i(x_1^c,\ldots,x_n^c)g_{\delta_1}(x_1^c)\cdots g_{\delta_n}(x_n^c)$ identically zero on $((\mathbb{Z}/q\mathbb{Z})^*)^n$.

Since the roots of $g_{\delta_1}(x_1^c)\cdots g_{\delta_n}(x_n^c)$ are a proper subset of $((\mathbb{Z}/q\mathbb{Z})^*)^n$, and since $\mathbb{Z}/q\mathbb{Z}$ has no zero divisors, we must have that for all i, some factor of h_i must have a root in $((\mathbb{Z}/q\mathbb{Z})^*)^n$. So we are done. ■

Our final algorithm is actually the simplest of the three algorithms of this paper.

Algorithm 5.3 (For `TorsionPoint` in general, assuming APH).
Input: *Polynomials* $f_1,\ldots,f_k \in \mathbb{Z}[x_1,\ldots,x_n]$, *positive integers* $d_1,\ldots,d_n$, *and a suitable value for the constant* $C\geq 1$ *from APH.*
Output: *A declaration of whether* $Z(f_1,\ldots,f_k)$ *contains a point* $\zeta=(\zeta_1,\ldots,\zeta_n)$ *with* $\zeta_i^{d_i}=1$ *for all* i*, meaningful and correct with probability* $>\frac{2}{3}$.
Description:

(1) *Let* $E := \max\{\max_i \deg f_i, \max_i d_i\}$, $M := \mathrm{lcm}\{d_1,\ldots,d_n\}$, *and let* $\sigma(F)$ *be one plus the maximum of the log of the absolute value of any coefficient of any* f_i.

(2) *Via recursive squaring, find the smallest* J, K, *and* L *such that* $L > 1+2(n+1)^3E^{n+1}(\sigma(F)+\log(k+n)+14(n+1)E\log(E+1))$, $K>\max\{e^{C^2}, 2^{\log^C M}, 36L^2\log^{2C}M\}$ *and* $J>\log(6)\log^C(KM)$.

(3) *Pick no more than* J *random* $j\in\{1,\ldots,K\}$ *until one either has* $q:=jM+1$ *prime, or* J *such numbers that are all composite. In the latter case, stop and output* `''I HAVE FAILED. PLEASE FORGIVE ME.''`.

(4) *Nondeterministically, decide whether the mod* q *reduction of*
$$f_1(x)=\cdots=f_k(x)=x_1^{d_1}-1=\cdots=x_n^{d_n}-1=0$$
has a root in $(Z/q\mathbb{Z})^n$.

(5) *If there is such a solution then stop and output* `''YES''`. *Otherwise, stop and output* `''NO''`. ◇

We are now ready to conclude the proof of Theorem 1.1.

Conclusion of Proof of Theorem 1.1: The equivalence of $\texttt{TorsionPoint}_1 \notin$ **P** and $\mathbf{P}\neq\mathbf{NP}$ follows immediately from our earlier remarks on the polynomial hierarchy [**Pap95**, Thm. 17.9], assuming we indeed have $\texttt{TorsionPoint}_1\in\mathbf{NP}^{\mathbf{NP}}$. The latter is contained in Assertion (2), which we already proved in the last section. So let us proceed with proving Assertions (1) and (3).

Proof of Assertion (1): It clearly suffices to show that Algorithm 5.3 is correct and runs within **AM**.

Let $F:=(f_1,\ldots,f_k)$. Correctness follows easily from Theorem 5.1 and Lemma 5.2. In particular, observe that K — the size of our sample space of numbers congruent to 1 mod M — is just large enough so that APH implies $\{1,\ldots,K\}$ contains at least $6L$ primes. (This follows easily from the basic implication $x\geq e^{C^2} \Longrightarrow \frac{x}{\log^C x}\geq\sqrt{x}$.) Notice also that if F does **not** vanish at any torsion point of interest, then the mod q reduction of F **does** vanish at a torsion point of interest for at most L primes q, thanks to Theorem 5.1. So the probability of a false `''YES''` answer is $<\frac{1}{6}$. Furthermore, by a routine binomial probability estimate, using the inequality $1-t\leq e^{-t}$ for $t\in(0,1)$, we obtain that the probability of drawing J composite integers is $<\frac{1}{6}$. In other words, with probability $>\frac{2}{3}$, Algorithm 5.3 gives the right answer.

To conclude, we need only observe that the seemingly large constants nevertheless yield low complexity. In particular, observe that the number of random bits necessary to do our random sampling is $O(J\log K)=O\Big([\log^C(M)+\log(L)]^{C+1}\Big)$ and the number of bit operations we must do is near-linear in $O(J\log K)$ (via fast finite field arithmetic [**BS96**, Ch. 5]). It is then easily checked that $\log M$ and $\log L$ (and thus J) are polynomial in our input size, so our algorithm is nothing more than a **BPP** algorithm, followed by a **single** call to an **NP**-oracle. This is exactly the definition of an **AM** algorithm [**BM88**], so we are done.

Proof of Assertion (3): Let us fix n and $d_1,\ldots,d_r$, and recall the notation of Algorithm 4.1 and the proof of Lemma 5.2. As observed in the proof of Lemma 5.2, F has a torsion point as specified iff there are positive integers $\delta_1,\ldots,\delta_n$ with $\delta_j|d_j$ for all j, such that for all i, the complex zero set of

$$\left(\prod_{\substack{(j_2,\ldots,j_n) \\ j_s \text{ co-prime to } \delta_s \forall s\in\{2,\ldots,n\}}} f_i(x_1,x_2^{j_2},\ldots,x_n^{j_n})\right) g_{\delta_1}(x_1)\cdots g_{\delta_n}(x_n)$$

contains $T(\delta_1e_1,\ldots,\delta_ne_n)$. By Lemma 2.3, since the number of factors and possible n-tuples $(\delta_1,\ldots,\delta_n)$ is constant, the preceding check can be done in **P**. ■

Example 5.4. *Consider the bivariate polynomial system* $F:=(f,g)$ *where* f *and* g *are respectively*

$$x^{3879876}y^{4594590}+x^{3879876}y^{4339335}+x^{3879876}y^{4084080}+x^{3879876}y^{3828825}+x^{2909907}y^{4594590}+$$
$$x^{3879876}y^{3573570}+x^{2909907}y^{4339335}+x^{3879876}y^{3318315}+x^{2909907}y^{4084080}+x^{3879876}y^{3063060}+$$
$$x^{2909907}y^{3828825}+x^{3879876}y^{2807805}+x^{1939938}y^{4594590}+x^{2909907}y^{3573570}+x^{3879876}y^{2552550}+$$
$$x^{1939938}y^{4339335}+x^{2909907}y^{3318315}+x^{3879876}y^{2297295}+x^{1939938}y^{4084080}+x^{2909907}y^{3063060}+$$
$$x^{3879876}y^{2042040}+x^{1939938}y^{3828825}+x^{2909907}y^{2807805}+x^{3879876}y^{1786785}+x^{969969}y^{4594590}+$$
$$x^{1939938}y^{3573570}+x^{2909907}y^{2552550}+x^{3879876}y^{1531530}+x^{969969}y^{4339335}+x^{1939938}y^{3318315}+$$
$$x^{2909907}y^{2297295}+x^{3879876}y^{1276275}+x^{969969}y^{4084080}+x^{1939938}y^{3063060}+x^{2909907}y^{2042040}+$$
$$x^{3879876}y^{1021020}+x^{969969}y^{3828825}+x^{1939938}y^{2807805}+x^{2909907}y^{1786785}+x^{3879876}y^{765765}+$$
$$y^{4594590}+x^{969969}y^{3573570}+x^{1939938}y^{2552550}+x^{2909907}y^{1531530}+x^{3879876}y^{510510}+y^{4339335}+$$
$$x^{969969}y^{3318315}+x^{1939938}y^{2297295}+x^{2909907}y^{1276275}+x^{3879876}y^{255255}+y^{4084080}+$$
$$x^{969969}y^{3063060}+x^{1939938}y^{2042040}+x^{2909907}y^{1021020}+x^{3879876}+y^{3828825}+x^{969969}y^{2807805}+$$
$$x^{1939938}y^{1786785}+x^{2909907}y^{765765}+y^{3573570}+x^{969969}y^{2552550}+x^{1939938}y^{1531530}+$$
$$x^{2909907}y^{510510}+y^{3318315}+x^{969969}y^{2297295}+x^{1939938}y^{1276275}+x^{2909907}y^{255255}+y^{3063060}+$$
$$x^{969969}y^{2042040}+x^{1939938}y^{1021020}+x^{2909907}+y^{2807805}+x^{969969}y^{1786785}+x^{1939938}y^{765765}+$$
$$y^{2552550}+x^{969969}y^{1531530}+x^{1939938}y^{510510}+y^{2297295}+x^{969969}y^{1276275}+x^{1939938}y^{255255}+$$
$$y^{2042040}+x^{969969}y^{1021020}+x^{1939938}+y^{1786785}+x^{969969}y^{765765}+y^{1531530}+x^{969969}y^{510510}+$$
$$y^{1276275}+x^{969969}y^{255255}+y^{1021020}+x^{969969}+y^{765765}+y^{510510}-x^{285285}+y^{255255}+2$$

and

$$x^{4594590}y^{285285}+x^{4339335}y^{285285}-x^{4594590}+x^{4084080}y^{285285}-x^{4339335}+x^{3828825}y^{285285}-$$
$$x^{4084080}-25\,y^{3879876}+x^{3573570}y^{285285}-x^{3828825}+x^{3318315}y^{285285}-x^{3573570}+x^{3063060}y^{285285}-$$
$$x^{3318315}+x^{2807805}y^{285285}-x^{3063060}-25\,y^{2909907}+x^{2552550}y^{285285}-x^{2807805}+x^{2297295}y^{285285}-$$
$$x^{2552550}+x^{2042040}y^{285285}-x^{2297295}+x^{1786785}y^{285285}-x^{2042040}-25\,y^{1939938}+x^{1531530}y^{285285}-$$
$$x^{1786785}+x^{1276275}y^{285285}-x^{1531530}+x^{1021020}y^{285285}-x^{1276275}+x^{765765}y^{285285}-x^{1021020}-$$
$$25\,y^{969969}+x^{510510}y^{285285}-x^{765765}+x^{255255}y^{285285}-x^{510510}+y^{285285}-x^{255255}-26,$$

which respectively have degrees 8474466 *and* 4879875, *and numbers of monomial* *terms* 96 *and* 42*. We would like to determine whether* F *vanishes at a point* (ζ, μ) *where both* ζ *and* μ *are* $4849845^{\underline{th}}$ *roots of unity.*

Algorithm 5.3 tells us we can do so, with a controllably small error probability, by finding a random prime q *of the form* $4849845c+1$ *and checking if the mod* q *reduction of* F *has a root in* $((\mathbb{Z}/q\mathbb{Z})^*)^2$*. Taking* $c=22$ *yields the prime* $q=106696591$*, and proceeding with this choice we see that the pair* $(75770298, 101629661)$ *is just such a root. This indicates that* F *may indeed vanish at a pair of* $4849845^{\underline{th}}$ *roots of unity, and running Algorithm 5.3* r *times would allow us to decide this with an error probability* $< \frac{1}{3^r}$ *by taking the answer that occupies the majority. (This example in fact vanishes at all* (ζ, μ) *with* ζ *and* μ ***primitive*** $95^{\underline{th}}$ *roots of unity so, since* $95|4849845$*, our putative answer is correct.)*

One could instead try to compute a Gröbner basis for the ideal $\langle f, g, x^{4849845}-1, y^{4849845}-1\rangle$*. The resulting basis will then be* $\{1\}$ *iff* F *does* **not** *vanish at any pair of* $4849845^{\underline{th}}$ *roots of unity. Trying one of the best Gröbner basis engines (*`Singular`*, version* `3-0-2`*), we are immediately thwarted: the maximum allowed exponent size is* 65536*. Trying three smaller examples with respective total degrees* 92114 *and* 65296 *(and respective numbers of monomial terms* 70 *and* 40*) resulted in* `‘‘Out of memory’’` *errors within about* 14 *minutes in all cases.*

On the other hand, while a brute-force implementation of Algorithm 5.3 can run slowly, the corresponding `Maple` *implementation has no memory problems for our examples here.* ◇

6. Is `TorsionPoint` NP-complete?

We close this paper by observing a possible speed-up to our last algorithm: One could instead simply attempt to nondeterministically guess a **small** number of suitable primes (instead of randomly sampling a large set), and then check nondeterministically whether one has torsion points modulo these primes. In particular, if the number of such "guessed" primes is polynomial in the input size, then it can be proved via the techniques of this paper that such an approach would yield `TorsionPoint`$\in$**NP**.

However, it is not clear how to prove that a small enough number of primes can be used. In particular, our final example shows that one definitely needs to use at least 3 primes, already for one variable.

Example 6.1. *Taking*

$$f(x_1):=4-3x_1^{10}-3x_1^{18}+4x_1^{42}-x_1^{60}+6x_1^{81}+5x_1^{95}-2x_1^{102}+3x_1^{105}$$

and $d:=210$*, it can be checked via* `Maple 9.5` *(within 2 hours, 13 minutes, and 42.63 seconds) that* f *does* **not** *vanish at any* $d^{\underline{th}}$ *root of unity. One would prefer to do this check modulo an intelligently chosen prime of the form* $210c+1$ *instead. However, there are exceptional primes which, using this approach, would cause one to falsely declare that* f **does** *vanish at a* $d^{\underline{th}}$ *root of unity. In this case it easily checked that the exceptional primes are exactly the divisors of the resultant of* f *and* $x^{210}-1$*, which (up to sign) is*

$$2227699600874096872564585144832612236369963246002360338615319497424201747782488174224095731882015016718028$$

and factors as

$$(2)^2(13)(29)(37)(43)(61)(71)(1801)(108557)(659101)(69529066111)(261727038763)(20353321490154047885351)(449182807853883447737046706077385 5421).$$

In particular, we see that the $11^{\underline{th}}$ *and* $14^{\underline{th}}$ *prime factors above are both congruent to* 1 *mod* 210*, and could thus lead to false* `‘‘YES’’` *answers.* ◇

It is interesting to note that Pascal Koiran has already given some evidence that it may be hard to prove that the more general problem $\texttt{FEAS}_{\mathbb{C}}$ is **NP**-complete. His evidence is based on the fact that $\texttt{FEAS}_{\mathbb{C}}$ contains a hard circuit-theoretic problem [**Koi96**, Sec. 6]. However, such a reduction does not appear to be known for `TorsionPoint`, so there may be more hope that $\texttt{TorsionPoint} \in \mathbf{NP}$ than $\texttt{FEAS}_{\mathbb{C}} \in \mathbf{NP}$.

Acknowledgements

This paper began while the author was attending the PIMS workshop on resolution of singularities, factorization of birational mappings, and toroidal geometry (December 11–16, 2004, Banff, Canada). The author thanks the organizers (Dan Abramovich, Ed Bierstone, Dale Cutkosky, Kenji Matsuki, Pierre Milman, and Jaroslaw Wlodarczyk) for their kind invitation. This paper was then presented at MEGA 2005 (Porto Conte, Alghero, Sardinia, Italy, June 1, 2005), and completed at Sandia National Laboratories. So the author thanks Patrizia Gianni, Tomas Recio, Carlo Traverso, Mark Danny Rintoul III, Philippe Pebay, and David Thompson for their generous invitations and wonderful choice of settings.

Special thanks go to Matt Baker for reading the ArXiV version of my paper the day it was posted and then pointing out the reference [**BZ95**]. Thanks also to Jeff Achter, Eric Bach, Alex Buium (who pointed out the reference [**Poo01**]), Petros Drineas, Sidney W. Graham, Bjorn Poonen, Rachel Pries, and Igor Shparlinski for some nice conversations and/or e-mail exchanges. I also thank an anonymous referee for pointing out Example 4.3, thus spotting an error in an earlier version of Theorem 1.1.

Finally, I would like to dedicate this paper to Helaman Ferguson, whom I had the honor of meeting at the January 2005 MAA-AMS-ASL joint meeting in Atlanta. Helaman: thank you for the π-disk!

References

[AKS04] Agrawal, Manindra; Kayal, Neeraj; and Saxena, Nitin, *"PRIMES is in P,"* Ann. of Math. (2) 160 (2004), no. 2, pp. 781–793.

[BM88] Babai, László and Moran, Shlomo, *"Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes,"* Journal of Computer and System Sciences, 36:254–276, 1988.

[BS96] Bach, Eric and Shallit, Jeff, *Algorithmic Number Theory, Vol. I: Efficient Algorithms,* MIT Press, Cambridge, MA, 1996.

[BPR06] Basu, Saugata; Pollack, Richard; and Roy, Marie-Françoise, *Algorithms in Real Algebraic Geometry,* Algorithms and Computation in Mathematics, vol. 10, 2nd ed., Springer-Verlag, 2006.

[BRS07] Bihan, Frederic; Rojas, J. Maurice; and Stella, Casey E., *"First Steps in Algorithmic Real Fewnomial Theory,"* submitted for publication. Downloadable from `http://www.math.tamu.edu/~rojas/list2.html` .

[BZ95] Bombieri, Enrico and Zannier, Umberto, *"Algebraic points on subvarieties of $\mathbf{G}^n_m$,"* Internat. Math. Res. Notices 1995, no. 7, pp. 333–347.

[Can88] Canny, John F., *"Some Algebraic and Geometric Computations in PSPACE,"* Proc. 20th ACM Symp. Theory of Computing, Chicago (1988), ACM Press.

[Cha38] Chabauty, C., *"Sur les équations diophantiennes liées aux unités d'un corps de nombres algébriques fini,"* Annali di Math. **17** (1938), pp. 127–168.

[CJ76] Conway, J. H. and Jones, A. J., *"Trigonometric diophantine equations (On vanishing sums of roots of unity),"* Acta Arith. 30 (1976), pp. 229–240.

[DN94] Deléglise, Marc and Nicolas, Jean-Louis, *"Sur les entiers inférieurs à x ayant plus de* $\log(x)$ *diviseurs,"* J. Théor. Nombres Bordeaux **6** (1994), no. 2, pp. 327–357.

[EP05] Emiris, Ioannis Z. and Pan, Victor Y., *"Improved Algorithms for Computing Determinants and Resultants,"* J. Complexity 21 (2005), no. 1, pp. 43–71.

[FS04] Filaseta, Michael and Schinzel, Andrzej, *"On testing the divisibility of lacunary polynomials by cyclotomic polynomials,"* Math. Comp. 73 (2004), no. 246, pp. 957–965.

[GJ79] Garey, Michael R. and Johnson, David S., *Computers and Intractability: A Guide to the Theory of NP-Completeness,* A Series of Books in the Mathematical Sciences, W. H. Freeman and Co., San Francisco, Calif., 1979.

[GKZ94] Gel'fand, Israel Moseyevitch; Kapranov, Misha M.; and Zelevinsky, Andrei V.; *Discriminants, Resultants and Multidimensional Determinants,* Birkhäuser, Boston, 1994.

[GKO96] Grigoriev, Dima Yu.; Karpinski, Marek; Odlyzko, Andrew, *"Short Proofs for Nondivisibility of Sparse Polynomials Under the Extended Riemann Hypothesis,"* Fund. Inform. **28** (1996), no. 3–4, pp. 297–301.

[Hal05] Hallgren, Sean, *"Fast quantum algorithms for computing the unit group and class group of a number field,"* STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 468–474, ACM, New York, 2005.

[HR17] Hardy, G. H. and Ramanujan, S., *"The normal number of prime factors of a number n,"* Quart. J. Math. 48 (1917), pp. 76–92.

[HB92] Heath-Brown, D. R., *"Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression,"* Proc. London Math. Soc. (3) 64 (1992), no. 2, pp. 265–338.

[Hil05] Hillar, Chris, *"Cyclic Resultants,"* Journal of Symbolic Computation, 39 (2005), pp. 653-669; erratum, ibid. 40 (2005), pp. 1126-1127.

[Ili89] Iliopoulos, Costas S., *"Worst Case Complexity Bounds on Algorithms for Computing the Canonical Structure of Finite Abelian Groups and the Hermite and Smith Normal Forms of an Integer Matrix,"* SIAM Journal on Computing, 18 (1989), no. 4, pp. 658–669.

[Koi96] Koiran, Pascal, *"Hilbert's Nullstellensatz is in the Polynomial Hierarchy,"* DIMACS Technical Report 96-27, July 1996. (This preprint considerably improves the published version which appeared Journal of Complexity **12** (1996), no. 4, pp. 273–286.)

[Koi97] Koiran, Pascal, *"Randomized and Deterministic Algorithms for the Dimension of Algebraic Varieties,"* Proceedings of the 38th Annual IEEE Computer Society Conference on Foundations of Computer Science (FOCS), Oct. 20–22, 1997, ACM Press.

[KPS01] Krick, Teresa; Pardo, Luis Miguel; and Sombra, Martín, *"Sharp estimates for the arithmetic Nullstellensatz,"* Duke Math. J. 109 (2001), no. 3, pp. 521–598.

[Lak91] Lakshman, Yagati N., *"A single exponential bound on the complexity of computing Gröbner bases of zero-dimensional ideals,"* Effective methods in algebraic geometry (Castiglioncello, 1990), pp. 227–234, Progr. Math., 94, Birkhäuser Boston, Boston, MA, 1991.

[Lau84] Laurent, Michel, *"Equations diophantiènnes exponentielles,"* Invent. Math. **78** (1984), pp. 299–327.

[Lan97] Lang, Serge, *Survey of Diophantine Geometry,* Springer-Verlag, 1997.

[McQ95] McQuillan, Michael, *"Division points on semi-abelian varieties,"* Invent. Math. 120 (1995), no. 1, pp. 143–159.

[Mih94] Mihailescu, Preda, *"Fast generation of provable primes using search in arithmetic progressions,"* Advances in cryptology — CRYPTO '94 (Santa Barbara, CA, 1994), pp. 282–293, Lecture Notes in Comput. Sci., 839, Springer, Berlin, 1994.

[Mil76] Miller, Gary L., *"Riemann's Hypothesis and Tests for Primality,"* J. Comput. System Sci. **13** (1976), no. 3, 300–317.

[MS01] Mulmuley, Ketan D. and Sohoni, Milind, *"Geometric complexity theory. I. An approach to the P vs. NP and related problems,"* SIAM J. Comput. 31 (2001), no. 2, pp. 496–526.

[NR83] Nicolas, J. L. and Robin, G., *"Majorations explicites pour le nombre de diviseurs de N,"* Canad. Math. Bull. **26** (1983), pp. 485–492.

[Pap95] Papadimitriou, Christos H., *Computational Complexity,* Addison-Wesley, 1995.

[Pla84] Plaisted, David A., *"New NP-Hard and NP-Complete Polynomial and Integer Divisibility Problems,"* Theoret. Comput. Sci. 31 (1984), no. 1–2, 125–138.

[Poo01] Poonen, Bjorn, *"Computing Torsion Points on Curves,"* Experiment. Math. **10** (2001), no. 3, pp. 449–465.

[Poo05] ____________, *e-mail communication,* November 6, 2005.

[Pra75] Pratt, Vaughan R., *"Every Prime has a Succinct Certificate,"* SIAM J. Comput. **4** (1975), 327–340.

[Pri06] Pries, Rachel, *conversation,* Fort Collins, Colorado, August 24, 2006.

[Roj01] Rojas, J. Maurice, *"Computational Arithmetic Geometry I: Diophantine Sentences Nearly Within the Polynomial Hierarchy,"* invited paper, Journal of Computer and System Sciences, vol. 62, no. 2, march 2001, pp. 216–235.

[Roj03] Rojas, J. Maurice, *"Dedekind Zeta Functions and the Complexity of Hilbert's Nullstellensatz"*, Math ArXiV preprint `math.NT/0301111`.

[Rup93] Ruppert, Wolfgang M., *"Solving algebraic equations in roots of unity,"* J. Reine Angew. Math. 435 (1993), pp. 119–156.

[Sch80] Schwartz, Jacob T., *"Fast Probabilistic Algorithms for Verification of Polynomial Identities,"* J. of the ACM 27, 701–717, 1980.

[Sma00] Smale, Steve, *"Mathematical Problems for the Next Century,"* Mathematics: Frontiers and Perspectives, pp. 271–294, Amer. Math. Soc., Providence, RI, 2000.

[Sto98] Storjohann, Arne, *"Computing Hermite and Smith normal forms of triangular integer matrices,"* Linear Algebra Appl. 282 (1998), no. 1–3, pp. 25–45.