Deterministic list decoding of Reed-Solomon codes

Soham Chatterjee* Prahladh Harsha* Mrinal Kumar*

Abstract

We show that Reed-Solomon codes of dimension k and block length n over any finite field $\mathbb{F}$ can be *deterministically* list decoded from agreement $\sqrt{(k-1)n}$ in time $\mathrm{poly}(n, \log|\mathbb{F}|)$.

Prior to this work, the list decoding algorithms for Reed-Solomon codes, from the celebrated results of Sudan and Guruswami-Sudan, were either randomized with time complexity $\mathrm{poly}(n, \log|\mathbb{F}|)$ or were deterministic with time complexity depending polynomially on the characteristic of the underlying field. In particular, over a prime field $\mathbb{F}$, no deterministic algorithms running in time $\mathrm{poly}(n, \log|\mathbb{F}|)$ were known for this problem.

Our main technical ingredient is a deterministic algorithm for solving the bivariate polynomial factorization instances that appear in the algorithm of Sudan and Guruswami-Sudan with only a $\mathrm{poly}(\log|\mathbb{F}|)$ dependence on the field size in its time complexity for every finite field $\mathbb{F}$. While the question of obtaining efficient deterministic algorithms for polynomial factorization over finite fields is a fundamental open problem even for *univariate* polynomials of degree 2, we show that additional information from the received word can be used to obtain such an algorithm for instances that appear in the course of list decoding Reed-Solomon codes.

Contents

*Tata Institute of Fundamental Research, Mumbai, India. {soham.chatterjee, prahladh, mrinal}@tifr.res.in. Research supported by the Department of Atomic Energy, Government of India, under project number RTI4014, and partially by reserach grants from SERB, Google research and Premji Invest.

1 Introduction

Reed–Solomon (RS) codes are among the most important and well-studied families of error-correcting codes, notable for their algebraic simplicity and optimal trade-off between rate and distance. Their simple algebraic structure lends to extremely efficient encoding and decoding algorithms, and RS codes are widely used both in theory and practice. An $[n, k]$ RS code over a finite field $\mathbb{F}$ encodes a message polynomial $f(X)$ of degree strictly less than k by evaluating it at n distinct field elements $\alpha_1, \ldots, \alpha_n$, yielding the codeword

$$(f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)) \in \mathbb{F}^n.$$

These codes attain the Singleton bound and have distance exactly $n - k + 1$.

Unique and list decoding. In the *unique decoding problem*, given a received word, the goal is to find the unique codeword, if it exists, within a Hamming ball of radius half the minimum distance, which for RS codes equals $(n - k + 1)/2$. Classical algorithms such as those of Berlekamp-Welch, Berlekamp-Massey and Peterson efficiently achieve this bound. However, when the corruption exceeds this radius, multiple codewords may have significant agreement with the received word. The *list decoding* problem, introduced by Elias [Eli57] and Wozencraft [Woz58], relaxes this require-

ment of uniqueness: instead of demanding uniqueness, the decoder outputs all codewords that are sufficiently close to the received word.

A sequence of breakthroughs, beginning with Sudan's algorithm [Sud97] and refined by Guruswami and Sudan [GS99], established that RS codes can be efficiently list decoded from agreement roughly $\sqrt{(k-1)n}$, corresponding to the Johnson radius, which is significantly larger than the half-the-minimum distance. These algorithms hinge on two key algebraic steps: (1) constructing a low-degree bivariate polynomial that vanishes at all received points with multiplicity, and (2) *factorizing* this polynomial to recover all potential message polynomials.

Derandomization and the factorization barrier. Both the Sudan and Guruswami–Sudan algorithms rely on the factorization of bivariate polynomials over finite fields. The known algorithms for polynomial factorization include algorithms for univariate polynomials due to Berlekamp [Ber67, Ber70] and Cantor–Zassenhaus [CZ81], and algorithms for multivariates developed in the 1980s (e.g. Kaltofen [Kal85], Lenstra [Len85]). These algorithms run in polynomial time[1] and are highly efficient in practice, but are unfortunately *randomized*. Despite decades of effort, obtaining efficient deterministic polynomial-time algorithms for polynomial factorization over finite fields[2] remains a fundamental open problem in computational algebra.

Even in the univariate case, where one seeks to factor a degree-d polynomial $f \in \mathbb{F}_q[X]$, all known deterministic algorithms either have running time polynomial in the *characteristic* of the field [Ber67, Ber70] or apply only to restricted classes of polynomials or work under unproven assumptions. For instance, while it is known that Shoup's improved deterministic factorization algorithm [Sho90] is efficient (i.e., takes time $\mathrm{poly}(d, |\log|\mathbb{F}|)$) for *most* polynomials, the best provable worst-case guarantee is $\sqrt{p} \cdot \mathrm{poly}(d, \log|\mathbb{F}|)$. Several deterministic algorithms based on the Generalized Riemann Hypothesis (GRH) have also been developed; however, these approaches not only rely on an unproven assumption but, like Shoup's algorithm, fail to handle *all* polynomials. In short, unconditional deterministic polynomial-time factorization over arbitrary finite fields remains elusive, even for quadratic polynomials.

This limitation directly impacts derandomization efforts in list decoding; since both Sudan and Guruswami–Sudan algorithms use the factorization step. As in the general polynomial factorization setting, all known deterministic algorithms for the list-decoding RS codes run in time polynomial in n and in the characteristic of the underlying field. Consequently, when the characteristic is polynomial in n (as is the case in most coding theoretic settings), these algorithms do yield a fully deterministic list-decoding procedure. However, no truly deterministic algorithm with running time polynomial in $(n, \log|\mathbb{F}|)$ was known for all settings of n and the finite field $\mathbb{F}$. In particular, even

[1] For the multivariate setting, the input and the output polynomials are generally succinctly represented by algebraic circuits.

[2] Over the field of rational numbers, deterministic polynomial time algorithms for univariate factorization were designed in the work of Lenstra, Lenstra & Lovasz [LLL82].

for prime fields $\mathbb{F}_p$ with $p \gg n$ (superpolynomial), the existence of such a deterministic algorithm remained open.

Our results. We give the first *deterministic list-decoding algorithms for RS codes* up to the Johnson radius that run in time polynomial in both the block length and logarithm of the field size. More precisely, we prove the following theorem.

Theorem 1.1. *There is a deterministic algorithm that for every finite field $\mathbb{F}$ and parameters $n > k \in \mathbb{N}$ runs in time $\mathrm{poly}(n, \log|\mathbb{F}|)$ and list decodes Reed-Solomon codes of block length n and dimension k over $\mathbb{F}$ from agreement greater than $\sqrt{(k-1)n}$.*

It would be interesting if we can obtain near-linear time versions of these deterministic algorithms (a la Alekhnovich [Ale05]).

Our work demonstrates that even though general deterministic factorization remains elusive, the algebraic structure inherent in coding-theoretic instances can circumvent this technical issue. This result is an addition to the broader *derandomization program* in computational complexity, offering yet another concrete setting where randomness can be provably removed without assuming any unproven hypotheses.

Techniques. Our main technical contribution is a new *deterministic algorithm for bivariate polynomial factorization* in the structured setting that arises within the algebraic list-decoding framework. While the general problem of deterministic factorization remains open, we exploit additional structure available from the received word, specifically, knowledge of its evaluation pattern and multiplicities, to remove randomness in this restricted but highly relevant setting.

This idea of exploiting additional information from the codeword also appears in the work of Kalev and Ta-Shma [KT22] in their analysis of a multiplicity-code-based lossless condenser. Inspired by their approach, we can obtain the following derandomization of Sudan's algorithm.

Theorem 1.2. *There is a deterministic algorithm that, for every finite field $\mathbb{F}$ and parameters $n, k \in \mathbb{N}$, runs in time $\mathrm{poly}(n, \log|\mathbb{F}|)$ and list decodes Reed–Solomon codes of block length n and dimension k over $\mathbb{F}$ from agreement greater than $\sqrt{2(k-1)n}$.*

Such a derandomization of Sudan's algorithm for Reed-Solomon codes (and their algebraic geometric variants due to Shokrollahi and Wasserman [SW99]) was discovered by Augot and Pecquet [AP00][3]. Our derandomized variant of Sudan's algorithm employs Newton's iteration[4] to find a root $(Y - f(X))$ of a bivariate polynomial $Q(X, Y)$. Typical applications of Newton's iteration use

[3] In an earlier version of this paper, we stated that such a derandomization was not previously known. We were unaware of the work of Augot and Pecquet [AP00] at that time. However, their approach is limited to the Sudan algorithm and does not extend to yield a derandomization of the Guruswami-Sudan algorithm, which is the main focus of our work.

[4] Kalev-TaShma used a similar approach to solve linear differential equations in [KT22].

a point (α, β) satisfying $Q(\alpha, \beta) = 0$ and $\frac{\partial Q}{\partial Y}(\alpha, \beta) \neq 0$. This point is typically obtained by choosing an appropriate α, and then using randomized univariate factorization algorithms to obtain a root β of $Q(\alpha, Y)$. To do this factorization deterministically in the list-decoding setting, we note that at all the points (α_j, β_j) in the received word, Q satisfies $Q(\alpha_j, \beta_j) = 0$. If furthermore, one of these points satisfy $\frac{\partial Q}{\partial Y}(\alpha, \beta) \neq 0$, we can avoid the *random* search for a suitable (α, β). Otherwise, if all points (α_j, β_j) of agreement of f with the codeword satisfy $\frac{\partial Q}{\partial Y}(\alpha, \beta) = 0$, we can use this to show that $(Y - f(X))$ is a root of $\frac{\partial Q}{\partial Y}(X, Y)$, a lower-degree polynomial. Applying this idea recursively, yields the above deterministic variant of Sudan's algorithm.

However, this idea alone is insufficient to derandomize the Guruswami–Sudan algorithm: in both Sudan's list decoding and Kalev–Ta-Shma's condenser, the multiplicity used in the interpolation step is $m = 1$, whereas in the Guruswami–Sudan algorithm the multiplicity is considerably higher.

We follow an alternate approach using Hensel lifting instead of Newton's iteration. Standard applications of Hensel lifting, like Newton's iteration, are seeded using an initial random step which we cannot afford. To circumvent this, we do the following: for each codeword point (α_j, β_j), we first deterministically factorize the interpolation polynomial $Q(\alpha_j, Y)$ into coprime factors as follows: $Q(\alpha_j, Y) = (Y - \beta_j)^{m_j} \cdot \widehat{P_j}(Y)$ for some $\widehat{P_j}(Y)$ such that $\widehat{P_j}(\beta_j) \neq 0$. We use this initial condition as a base case for the Hensel lifting and obtain a non-trivial local factorization of Q corresponding to each point (α_j, β_j) in the received word. We then apply this local factorization recursively to the set of factors obtained, till the set stabilizes. We then show how to obtain the desired factors $(Y - f(X))$ corresponding to the polynomials f in the list from the stabilized factorization.

A detailed overview of these deterministic algorithms is presented in Section 2.

2 Proof overview

In this section, we outline the main ideas behind the proof of Theorem 1.1. To set the stage, we first recall the structure of the list-decoding algorithms of Sudan [Sud97] and Guruswami–Sudan [GS99]. Let $w = \{(\alpha_j, \beta_j)\}_{j \in [n]}$ denote the received word, where $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ are distinct evaluation points. The algorithms depend on parameters m, D, t, to be fixed later, and output a list L of degree-$(k-1)$ univariate polynomials $f(X)$ that agree with w on at least t points. Both algorithms share the same two-step structure:

1. **Interpolation.** Find a nonzero polynomial $Q(X, Y)$ of $(1, k-1)$-degree[5] at most D that vanishes at each (α_j, β_j) with multiplicity at least m. (Formally, all Hasse derivatives of order less than m vanish at each point.)

2. **Factorization.** Factor $Q(X, Y)$ over $\mathbb{F}$; for each factor of the form $(Y - f(X))$, output f if $\deg f < k$ and f agrees with w on at least t positions.

[5]The (a, b)-degree of a monomial $x^i y^j$ is defined to be equal to $(ai + bj)$. The (a, b)-degree of a polynomial Q is the maximum of the (a, b)-degrees of the non-zero monomials in the polynomial.

The distinction between the two algorithms lies in the choice of parameters. Sudan's algorithm sets $m = 1$ and $D, t \approx \sqrt{2(k-1)n}$, while Guruswami–Sudan sets $m = \mathrm{poly}(n)$ to decode from agreement $t = \sqrt{(k-1)n}$, with $D \approx m\sqrt{(k-1)n}$. The use of higher multiplicities, one of the key innovations of [GS99], has since become a standard tool in algebraic coding theory [Sar11].

The interpolation step reduces to solving a homogeneous linear system in the coefficients of Q. The parameters are chosen so that the number of constraints is smaller than the number of variables and hence a nonzero solution always exists and can be found deterministically in $\mathrm{poly}(n, \log q)$ time by Gaussian elimination.

Randomness enters only in the *factorization* step, which employs standard algorithms for bivariate polynomial factorization over $\mathbb{F}$. A natural approach to derandomizing the decoding algorithm would be to design a deterministic $\mathrm{poly}(d, \log |\mathbb{F}|)$-time algorithm for factoring arbitrary bivariate polynomials of degree d over a finite field $\mathbb{F}$. However, such an algorithm is unknown even for univariate quadratics. The factorization instances that arise in the decoding algorithms of [Sud97, GS99] exhibit additional structure: the desired factors are always linear in Y (of the form $(Y - f(X))$ for a univariate f of degree less than k), and we have access to noisy evaluations of f. We show that this additional structure can be exploited to eliminate randomness efficiently.

The techniques differ slightly between the two parameter settings, and we first describe the simpler case corresponding to Sudan's algorithm before turning to the Guruswami–Sudan algorithm.

2.1 Derandomizing Sudan's decoder

As mentioned in the introduction, a derandomization of Sudan's algorithm was discovered earlier by Augot and Pecquet [AP00] using a Hensel lifting approach. We find it constructive to recall their result in our framework using Newton's iteration, as it provides a natural stepping stone toward the more general derandomization of the Guruswami-Sudan decoder. In Sudan's algorithm, the interpolation step outputs a polynomial $Q(X, Y)$ that vanishes on each (α_j, β_j) and has $(1, k-1)$-degree less than $\sqrt{2(k-1)n}$. The analysis observes that for any polynomial $f(X)$, the univariate $Q(X, f(X))$ vanishes at every α_j where $f(\alpha_j) = \beta_j$. Hence, if the agreement between f and the received word exceeds the degree of $Q(X, f(X))$ (bounded by the $(1, k-1)$-degree of Q), then $Q(X, f(X))$ must be identically zero or equivalently, $f(X)$ is a Y-root of $Q(X, Y)$. To recover such an f from Q, we employ the classical Newton iteration technique, stated below.

Lemma 2.1 (Newton Iteration [GG13, Chapter 9])**.** *Let $\mathcal{R} = \mathbb{F}[X]$ be a polynomial ring and let $Q(X, Y) \in \mathcal{R}[Y]$. Suppose $\varphi \in \mathbb{F}[\![X]\!]$ satisfies $Q(X, \varphi) \equiv 0 \bmod \langle X \rangle^m$ and $Q^{(0,1)}(0, \varphi(0)) \neq 0$. Then*

$$\varphi' := \varphi - \frac{Q(X, \varphi)}{Q^{(0,1)}(0, \varphi(0))}$$

satisfies $Q(X, \varphi') \equiv 0 \bmod \langle X \rangle^{m+1}$ and $\varphi' \equiv \varphi \bmod \langle X \rangle^m$. Moreover, this extension is unique: any

φ'' satisfying $Q(X, \varphi'') \equiv 0 \bmod \langle X \rangle^{m+1}$ and $\varphi'' \equiv \varphi \bmod \langle X \rangle^m$ must also satisfy $\varphi'' \equiv \varphi' \bmod \langle X \rangle^m$.

In our setting, the lemma implies that if we can find $(\alpha, \beta) \in \mathbb{F}^2$ such that $f(\alpha) = \beta$ and $Q^{(0,1)}(\alpha, \beta) \neq 0$, then starting from

$$Q(X, \beta) \equiv 0 \bmod \langle X - \alpha \rangle,$$

we can iteratively lift modulo powers of $\langle X - \alpha \rangle$ using Lemma 2.1 to recover f. This process is fully deterministic and can be implemented in $\mathrm{poly}(\deg Q, \log |\mathbb{F}|)$ time over any finite field. Thus, recovering f reduces to finding a point (α, β) such that $f(\alpha) = \beta$ and $Q^{(0,1)}(\alpha, \beta) \neq 0$.

Given the received word w, there are many candidate points (α_j, β_j) where $f(\alpha_j) = \beta_j$. Although we do not know which coordinates correspond to actual agreements, we can simply try all n possibilities. If some j satisfies $Q^{(0,1)}(\alpha_j, \beta_j) \neq 0$, then we can deterministically recover f via Lemma 2.1.

The remaining case is for those close enough polynomials f such that for every point of agreement (α_j, β_j), the derivative $Q^{(0,1)}(\alpha_j, \beta_j) = 0$. In this situation, f is in fact a Y-root of $Q^{(0,1)}(X, Y)$. The Y-degree of $Q^{(0,1)}$ is strictly smaller than that of Q, while its $(1, k-1)$-degree does not increase in the process. Define $R_1(X) = Q^{(0,1)}(X, f(X))$. By assumption, $R_1(\alpha_j) = 0$ at all points of agreement of f, implying $R_1(X) \equiv 0$. Hence, $f(X)$ is also a Y-root of $Q^{(0,1)}(X, Y)$.

We now repeat the same reasoning with $Q^{(0,1)}$ in place of Q. If there exists (α_j, β_j) satisfying the conditions of Lemma 2.1, we recover f; otherwise, f must also be a Y-root of $Q^{(0,2)}(X, Y)$, and so on. Consequently, the decoder simply tries all partial derivatives $Q^{(0,i)}(X, Y)$ for successive i and all candidate points (α_j, β_j). For each pair, it invokes the Newton iteration subroutine. Correctness follows from the observation that for every codeword f close enough to w, at least one such invocation succeeds. The technical details are deferred to Section 4.

2.2 Derandomizing the Guruswami–Sudan decoder

Given the previous section, it is natural to ask whether iterating over the Y-derivatives of Q also works in the Guruswami–Sudan parameter regime. This direct approach fails. To decode from agreement $\sqrt{(k-1)n}$, the interpolation multiplicity is set to $m = \mathrm{poly}(n)$, so Q vanishes with multiplicity at least m at each (α_j, β_j). Consequently, for any $f(X)$ agreeing with w at (α_j, β_j), the univariate $Q(X, f(X))$ vanishes with multiplicity at least m at $X = \alpha_j$, and hence we can conclude that $Q(X, f(X))$ is identically zero if

$$m \cdot \#\text{agreements}(f, w) > (1, k-1)\text{-degree of } Q. \tag{2.2}$$

One might try to recover f by Newton iteration from an agreement point. However, since $m > 1$, we have $Q^{(0,1)}(\alpha_j, \beta_j) = 0$ for every (α_j, β_j); thus we can attempt to recurse on $Q^{(0,1)}$. For this to

work, we would need

$$(m-1)\cdot \#\text{agreements}(f,w) > (1,k-1)\text{-degree of } Q^{(0,1)}. \tag{2.3}$$

But $(1,k-1)$-degree of $Q^{(0,1)}$ can be as large as $D-k$ where $D = (1,k-1)$-degree of Q, so relative to (2.2) the left-hand side drops by at least $\sqrt{(k-1)n}$ while the right-hand side drops by only k. Thus (2.3) need not hold, and the simple derivative recursion from the Sudan case does not extend.

To overcome this, we refine Q via a local factor-type decomposition at each received point. Let $P = \prod_{i=1}^{s} P_i$ be the factorization of a *monic* bivariate $P(X,Y) \in \mathbb{F}[X,Y]$ into irreducibles (with multiplicity). For simplicity, we will assume in this proof overview section that all the bivariates are monic in the variable Y. While the monic setting conveys most of the high level ideas of the proof, for our final proofs, we have to work with non-monic bivariates and this leads to some additional technical difficulties that we handle in Section 5 and Section 6. For $(\alpha,\beta) \in \mathbb{F}^2$, we partition the set $[s]$ corresponding to the factors of P, into 3 sets as follows:

$$\begin{aligned}
A(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha,Y) = \gamma\,(Y-\beta)^{m_i} \text{ for some } m_i \geq 1,\ \gamma \in \mathbb{F}\setminus\{0\}\},\\
B(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha,\beta) \neq 0, \deg_Y P_i \geq 1\},\\
C(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha,Y) = (Y-\beta)^{m_i}\cdot \widehat{P}_i(Y) \text{ for some } m_i \geq 1, \deg_Y \widehat{P}_i \geq 1, \widehat{P}_i(\beta) \neq 0\}.
\end{aligned}$$

Let $P_{A(\alpha,\beta)} := \prod_{i\in A(\alpha,\beta)} P_i$ and $P_{B(\alpha,\beta)}$, $P_{C(\alpha,\beta)}$ be defined similarly. For the received word $w = (\alpha_j,\beta_j)_{j\in[n]}$, write A_j, B_j, C_j and $P_{A,j}, P_{B,j}, P_{C,j}$ instead of $A(\alpha_j,\beta_j), B(\alpha_j,\beta_j), C(\alpha_j,\beta_j)$ and $P_{A(\alpha_j,\beta_j)}, P_{B(\alpha_j,\beta_j)}, P_{C(\alpha_j,\beta_j)}$ for (α_j,β_j).

Since P is monic in Y, it follows from the Gauss' lemma that each of its irreducible factors can be assumed to be monic in Y. And, thus, P must equal the product of $P_{A,j}, P_{B,j}$ and $P_{C,j}$.

A simple but key observation is the following when applied to the Guruswami–Sudan interpolation polynomial Q.

Observation 2.4. *Let Q be the Guruswami–Sudan interpolation polynomial and $f \in L$. Then:*

1. *If $f(\alpha_j) = \beta_j$ for some j, then $(Y - f(X))$ divides $Q_{A,j}$.*
2. *If $Q = Q_{A,j}$ for some j, then $f(\alpha_j) = \beta_j$.*
3. *If $f(\alpha_j) \neq \beta_j$ for some j, then $(Y - f(X))$ divides $Q_{B,j}$*

The first item above follows from the definition of $Q_{A,j}$ and similarly the third item from the definition of $Q_{B,j}$. For the second item, we recall that $(Y - f(X))$ divides $Q(X,Y)$ for every f in the list L. As a consequence, $(Y - f(\alpha_j))$ must divide $Q(\alpha_j, Y)$, which by the definition of Q_{A_j} and the fact that $Q = Q_{A,j}$, only has irreducible factors of the form $(Y - \beta_j)$. So, $f(\alpha_j)$ must equal β_j.

Thus, if we can produce a bivariate P divisible by $(Y - f(X))$ and satisfying $P = P_{A,j}$ for at least k indices j, then we can recover f by interpolating on those (α_j,β_j).

Our algorithm (Theorem 1.1) is an efficient deterministic realization of this plan, built from three components.

(1) Deterministic local splitting $P \mapsto (P_{A,j}, P_{B,j}, P_{C,j})$.

Theorem 2.5 (monic version of Theorem 5.8)**.** *Let $\mathbb{F}$ be any finite field. There is a deterministic algorithm* Split *which, given a polynomial $P \in \mathbb{F}[X,Y]$ that is monic in the variable Y and a point $(\alpha,\beta) \in \mathbb{F}^2$, outputs polynomials $P_{A(\alpha,\beta)}$, $P_{B(\alpha,\beta)}$ in time* $\mathrm{poly}(\deg P, \log|\mathbb{F}|)$.

The method follows the Hensel-lifting paradigm but avoids randomized univariate factorization by using (α_j, β_j) directly. First write

$$P(\alpha_j, Y) = (Y-\beta_j)^{m_j} \cdot \widehat{P}_j(Y), \qquad \text{where } \widehat{P}_j(\beta_j) \neq 0,$$

computed deterministically by repeated division by $(Y-\beta_j)$. If $m_j = 0$ then $P = P_{B,j}$; if $m_j = \deg_Y P(\alpha_j, Y)$ then $P = P_{A,j}$; otherwise the coprime factorization $\big((Y-\beta_j)^{m_j}, \widehat{P}_j(Y)\big)$ serves as the base case for Hensel lifting to recover $P_{A,j}, P_{B,j}, P_{C,j}$ deterministically. While the standard application of Hensel lifting (c.f., [KSS15, ST21]) uses randomized univariate factorization to seed it, the above deterministic factoring of $P(\alpha_j, Y)$ into coprime factors $(Y-\beta_j)^{m_j}$ and $\widehat{P_j}(Y)$ circumvents the use of randomness in our application of Hensel lifting. See Section 6 for details.

(2) Refinement to a stable set. Call P *stable* (with respect to the received word w) if for every $j \in [n]$, either $P = P_{A,j}$ or $P = P_{B,j}$. Initialize $S \leftarrow \{Q\}$. While some $P \in S$ is not stable (i.e., there is a $j \in [n]$ such that $P \notin \{P_{A_j}, P_{B_j}\}$), update

$$S \leftarrow \Big(S \cup \{P_{A,j}, P_{B,j}\}\Big) \setminus \{P\}.$$

This refinement uses Theorem 2.5 at each step.

(3) Decoding from stability and efficiency. If S is stable and $(Y-f) \mid P \in S$, then by Observation 2.4, every j with $P = P_{A,j}$ corresponds to a true agreement $f(\alpha_j) = \beta_j$. We prove that after stabilization, for every $f \in L$ there exists $P \in S$ such that $(Y-f) \mid P$ and $P = P_{A,j}$ holds on at least k indices j. Interpolating on these points recovers f, and iterating over $P \in S$ outputs all $f \in L$.

To conclude the argument, we need to show that the refinement to a stable set can be done in polynomial time. We do this via a potential function argument. See Section 5 for the full details and analysis.

3 Preliminaries

For a natural number n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We say that a function $T(n)$ is poly(n) if there is a constant c (independent of n) such that for all sufficiently large n, $T(n) \leq n^c$.

Throughout the paper, $\mathbb{F}$ denotes a finite field of size $q = p^r$, for some prime p and positive integer r. The prime p is referred to as *characteristic* of the field $\mathbb{F}$. We will be working with the ring $\mathbb{F}[X, Y]$ of bivariate polynomials over the field $\mathbb{F}$.

3.1 Resultants and GCD

Greatest common divisor: The *greatest common divisor (GCD)* of two polynomials $P, Q \in \mathbb{F}[X, Y]$, denoted by $\gcd(P, Q)$ is a polynomial $D \in \mathbb{F}[X, Y]$ that divides P and Q, and such that every common divisor of P and Q also divides D. If $D = \gcd(P, Q)$, there exist two polynomials $A, B \in \mathbb{F}[X, Y]$, referred to as the *Bezout coefficients* such that

$$A \cdot P + B \cdot Q = D.$$

Resultant: Let $g = \sum_{i=0}^{d_1} g_i(X)Y^i$ and $h = \sum_{i=0}^{d_2} h_i(X)Y^i$ be polynomials in $\mathbb{F}[X, Y]$ of Y-degree exactly d_1 and d_2. Denote the vector space $W_d = \{f \in \mathbb{F}[X, Y] \mid \deg_Y(f) < d\}$ over $\mathbb{F}(X)$. Then consider linear map $\psi : W_{d_2} \times W_{d_1} \to W_{d_1+d_2}$ where $\psi(a(X, Y), b(X, Y)) = a(X, Y) \cdot g(X, Y) + b(X, Y) \cdot h(X, Y)$ where $a, b \in \mathbb{F}[X, Y]$ and $\deg_Y(a) < d_2$, $\deg_Y(b) < d_1$. Then the matrix of the ψ, over $\mathbb{F}[X]$ is called the *Sylvester Matrix* with respect to Y. And the determinant the Sylvester matrix is called the *Resultant* of g, h with respect to Y, denoted by $\mathrm{Res}_Y(g, h)$

Lemma 3.1 (Resultant Properties)**.** *Let $d_1, d_2 \geq 1$. Let $g(X, Y), h(X, Y) \in \mathbb{F}[X, Y]$ have Y-degree exactly d_1, d_2 respectively. Then the following are true:*

1. *If each g_i is a polynomial of degree at most e_1 and each h_j is a polynomial of degree at most e_2 then $\deg(\mathrm{Res}_Y(g, h)) \leq 2e_1e_2$*
2. *$\mathrm{Res}_Y(g, h) = 0$ if and only if g, h have a nontrivial GCD in the ring $\mathbb{F}(X)[Y]$.*
3. *There exist polynomials $a(X, Y)$ and $b(X, Y)$ in $\mathbb{F}[X, Y]$ such that:*

$$\mathrm{Res}_Y(g, h) = a \cdot g + b \cdot h$$

The following lemma is an easy consequence of Gauss' lemma and will be useful for us.

Lemma 3.2 (Gauss' lemma)**.** *Let $A(X, Y) = \sum_{i=0}^{d} A_i(X)Y^i$ and $B(X, Y) = \sum_{j=0}^{e} B_j(X)Y^j$ be polynomials in $\mathbb{F}[X, Y]$ of Y-degrees at least one.*

Then, A and B have a GCD with Y-degree at least one in the ring $\mathbb{F}(X)[Y]$ if and only if they have GCD with Y-degree at least one in $\mathbb{F}[X, Y]$.

3.2 Hasse derivatives and multiplicties

In many parts of the proof, especially in the context of the list decoding algorithm of Guruswami–Sudan, we have to work with the notion of multiplicities. We start by recalling the notion of Hasse derivatives and multiplicity.

Definition 3.3 (Hasse derivatives)**.** *Let $Q(X,Y) \in \mathbb{F}[X,Y]$ be a bivariate polynomial. Then for any $(e_1,e_2) \in \mathbb{Z}^2_{\geq 0}$, the (e_1,e_2)-th Hasse derivative of Q, denoted by $Q^{(e_1,e_2)}(X,Y)$ is defined to be the coefficient of the monomial $Z_1^{e_1}Z_2^{e_2}$ in the polynomial $Q(X+Z_1,Y+Z_2)$, when it is viewed as a bivariate in Z_1,Z_2.*

(e_1+e_2) is said to be the order *of the Hasse derivative $Q^{(e_1,e_2)}$.*

$\Diamond$

Definition 3.4 (multiplicity)**.** *Let $Q(X,Y) \in \mathbb{F}[X,Y]$ be a bivariate polynomial, m be a non-negative integer and $(\alpha,\beta) \in \mathbb{F}^2$ be a point. We say that Q vanishes with multiplicity at least m on on (α,β) if every Hasse derivative of Q of order at most $(m-1)$ evaluates to zero at (α,β).* $\Diamond$

The definitions of both Hasse derivatives and multiplicity extend naturally to polynomials with an arbitrary number of variables. Throughout the paper, these are invoked only for bivariate or univariate polynomials.

3.3 Computing with polynomials:

In this section, we mention several standard *deterministic* building blocks for working with polynomials.

Exact division: There exists a deterministic algorithm that on input two polynomials $P,Q \in \mathbb{F}[X,Y]$ with the guarantee that Q divides P, finds the quotient P/Q in time $\mathrm{poly}(\deg P, \deg Q, \log|\mathbb{F}|)$.

GCD Computation: Euclid's algorithm computes the GCD of n univariate polynomials $P_1,\ldots,P_n \in \mathbb{F}[X]$ in time $\mathrm{poly}(\max \deg P_i, \log|\mathbb{F}|)$. The extended-Euclid algorithm, on input two polynomials $P,Q \in \mathbb{F}[X]$ computes the $\gcd(P,Q)$ as well as the Bezout coeffients in $\mathrm{poly}(\deg P, \deg Q, \log|\mathbb{F}|)$ time.

Content of polynomials. Given a polynomial $P \in \mathbb{F}[X,Y]$, we can view it as a polynomial in Y with coefficients from the ring $\mathbb{F}[X]$, more precisely write $P(X,Y) = \sum_\ell P^{(\ell)}(X)\cdot Y^\ell$. The X-content of the polynomial P is $\gcd(P^{(1)}, P^{(2)}, \ldots)$.

Polynomial factorization for small characteristic fields: As mentioned in the introduction, Shoup's algorithm [Sho90] computes the factorization of degree d bivariate polynomials in deterministic $\sqrt{p}\cdot\mathrm{poly}(d,\log|\mathbb{F}|)$ time where p is the characteristic of the field.

For fields of larger characteristic, we will use the following standard tools from computational algebra: Newton's iteration and Hensel lifting.

Newton's iteration. As mentioned in the proof overview (Section 2), we will be using Newton's iteration repeatedly to find the Y-roots of bivariate polynomials.

Lemma 2.1 (Newton Iteration [GG13, Chapter 9])**.** *Let $\mathcal{R} = \mathbb{F}[X]$ be a polynomial ring and let $Q(X,Y) \in \mathcal{R}[Y]$. Suppose $\varphi \in \mathbb{F}[\![X]\!]$ satisfies $Q(X,\varphi) \equiv 0 \bmod \langle X\rangle^m$ and $Q^{(0,1)}(0,\varphi(0)) \neq 0$. Then*

$$\varphi' := \varphi - \frac{Q(X,\varphi)}{Q^{(0,1)}(0,\varphi(0))}$$

satisfies $Q(X,\varphi') \equiv 0 \bmod \langle X\rangle^{m+1}$ and $\varphi' \equiv \varphi \bmod \langle X\rangle^m$. Moreover, this extension is unique: any φ'' satisfying $Q(X,\varphi'') \equiv 0 \bmod \langle X\rangle^{m+1}$ and $\varphi'' \equiv \varphi \bmod \langle X\rangle^m$ must also satisfy $\varphi'' \equiv \varphi' \bmod \langle X\rangle^m$.

Hensel lifting. Hensel lifting allows us to lift a local factorization of a polynomial modulo a power of $(X-\alpha)$ to a higher power. This may be viewed as the analog of Newton iteration for factoring, as opposed to root lifting. We give more details of the Hensel lifting in Section 6.1.

A note on the characteristic of the underlying field. Typical applications of Newton's iteration and Hensel's lifting assume that the characteristic of the underlying field is at least the degree of the polynomials involved. While it is possible to use these ideas for polynomial factorization over fields of small characteristic with some care, it will be convenient for us to assume that the characteristic is at least $\mathrm{poly}(n)$ for our applications, since anyway when the characteristic is smaller one can use Shoup's deterministic factorization to obtain deterministic list-decoding algorithms.

4 Derandomizing Sudan's decoder

In this section, we prove Theorem 1.2. Recall that Sudan's list-decoding algorithm consists of two main steps:

1. **Interpolation Step:** Given the received word $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$, find a nonzero bivariate polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$ of $(1,k-1)$-degree at most D such that $Q(\alpha_j,\beta_j) = 0$ for all $j \in [n]$. Here D is a parameter depending on n and k, set to $D = \sqrt{2(k-1)n}$.
2. **Factorization Step:** Find all factors of $Q(X,Y)$ of the form $Y - f(X)$ where $\deg(f) \leq k-1$, such that f agrees with w on more than $\sqrt{2(k-1)n}$ coordinates.

The first step can be carried out deterministically in polynomial time by solving a homogeneous linear system: there are more variables than equations, guaranteeing a nontrivial solution. Thus, the only source of randomness lies in the *factorization step*. The main technical result of this section is the following theorem, which, combined with the above observation, immediately implies Theorem 1.2.

Theorem 4.1. *There is a deterministic algorithm that, for every finite field $\mathbb{F}$ and parameters $n, k \in \mathbb{N}$, runs in time $\mathrm{poly}(n, \log|\mathbb{F}|)$ and, given as input a received word $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$ and a polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$ of $(1, k-1)$-degree at most $\sqrt{2(k-1)n}$ satisfying $Q(\alpha_j, \beta_j) = 0$ for all $j \in [n]$, returns all polynomials $f \in \mathbb{F}[X]$ of degree at most $(k-1)$ that agree with w on more than $\sqrt{2(k-1)n}$ coordinates.*

We now describe and analyze the deterministic algorithm.

4.1 The algorithm

Algorithm 1: DETERMINISTIC-SUDAN

Input: $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$ and $Q(X,Y) \in \mathbb{F}[X,Y]$ of $(1,k-1)$-degree at most $\sqrt{2(k-1)n}$ such that $Q(\alpha_j, \beta_j) = 0$ for all $j \in [n]$

Output: All $f \in \mathbb{F}[X]$ of degree at most $(k-1)$ that agree with w on more than $\sqrt{2(k-1)n}$ coordinates

---

if *there exists $f(X)$ of degree $\le k-1$ such that $f(\alpha_j) = \beta_j$ for all $j \in [n]$* **then return** $\{f\}$
if $\deg_Y(Q) = 1$ **then**
  Write $Q(X,Y) = Q_0(X) + Y \cdot Q_1(X)$
  Set $f(X) \leftarrow -\frac{Q_0(X)}{Q_1(X)}$
  if $\deg(f) \le k-1$ *and f has agreement with w exceeding* $\sqrt{2(k-1)n}$ **then return** $\{f\}$
Set $L \leftarrow \emptyset$
Set $P \leftarrow Q^{(0,1)}(X,Y)$
while $\deg_Y(P) > 0$ *and there exists $j \in [n]$ such that $P(\alpha_j, \beta_j) \neq 0$* **do**
  $\varphi_0 \leftarrow \beta_j$
  for $i = 1$ *to* $(k-1)$ **do**
    Use Newton Iteration (Lemma 2.1) to compute the lift φ_i from φ_{i-1} satisfying $Q(X, \varphi_i(X)) \equiv 0 \pmod{\langle X - \alpha_j\rangle^{i+1}}$ and $\varphi_i \equiv \varphi_{i-1} \pmod{\langle X - \alpha_j\rangle^{i}}$
  Set $\varphi \leftarrow \varphi_{k-1} \bmod \langle X - \alpha_j\rangle^k$
  if φ *agrees with w on more than* $\sqrt{2(k-1)n}$ *coordinates* **then** $L \leftarrow L \cup \{\varphi\}$
  Set $w \leftarrow w \setminus \{(\alpha_j, \beta_j)\}$
$L' \leftarrow$ DETERMINISTIC-SUDAN(P, w)
return $L \cup L'$

4.2 Correctness of the algorithm

The theorem is interesting only when the desired agreement threshold, $\sqrt{2(k-1)n}+1$, is smaller than n. If it equals n, the problem reduces to simple polynomial interpolation: we can directly check whether there exists f of degree at most $k-1$ such that $f(\alpha_j)=\beta_j$ for all $j \in [n]$.

Henceforth, assume $\deg_Y(Q) \geq 1$. If $\deg_Y(Q)=0$, then Q is a univariate polynomial in X of degree at most $\sqrt{2(k-1)n}$ vanishing at n distinct points, which would imply $Q \equiv 0$, a trivial case.

We begin with a simple observation.

Claim 4.2. *Let $f(X)$ be a polynomial of degree $< k$ that agrees with w on more than $\sqrt{2(k-1)n}$ coordinates. Let $R(X,Y)$ be a nonzero polynomial of $(1,k-1)$-degree at most $\sqrt{2(k-1)n}$ such that $R(\alpha_j,\beta_j)=0$ for every j where $f(\alpha_j)=\beta_j$. Then $R(X,f(X))$ is identically zero.*

Proof. The univariate $R(X,f(X))$ has degree at most the $(1,k-1)$-degree of R, namely $\sqrt{2(k-1)n}$. It vanishes at each α_j where $f(\alpha_j)=\beta_j$, i.e., on more than $\sqrt{2(k-1)n}$ distinct points. Hence $R(X,f(X))$ has more zeros than its degree and must be identically zero. □

By Claim 4.2, for any such f, we have $Q(X,f(X)) \equiv 0$. We now prove by induction on $\deg_Y(Q)$ that every such f is included in the list L.

Base case: If $\deg_Y(Q)=1$, there is at most one polynomial f such that $Q(X,f(X)) \equiv 0$, and the algorithm explicitly checks for this case.

Induction step: Assume correctness for all polynomials of Y-degree at most t. Let Q have Y-degree $t+1>1$. We consider two cases:

Case I: There exists j such that $f(\alpha_j)=\beta_j$ and $Q^{(0,1)}(\alpha_j,\beta_j) \neq 0$. Then the conditions of the Newton Iteration lemma (Lemma 2.1) are met, and the algorithm successfully lifts β_j modulo successive powers of $\langle X-\alpha_j \rangle$ to recover f. Since f has sufficiently large agreement, it is added to L.

Case II: For every j with $f(\alpha_j)=\beta_j$, we also have $Q^{(0,1)}(\alpha_j,\beta_j)=0$. In this case, $Q^{(0,1)}(X,Y)$ is a nonzero polynomial (since $\deg_Y(Q)>1$) of $(1,k-1)$-degree at most $\sqrt{2(k-1)n}$ that vanishes on all agreement points of f. By Claim 4.2, $Q^{(0,1)}(X,f(X)) \equiv 0$, so f is a Y-root of $Q^{(0,1)}(X,Y)$. By the induction hypothesis, the recursive call with $Q^{(0,1)}$ includes f in L.

This completes the proof of correctness.

4.3 Running time analysis

In each recursive call, the Y-degree of Q decreases by at least one. Hence, the recursion depth is at most $\deg_Y(Q)=O(\sqrt{2(k-1)n})$, and only one recursive call is made per level. Each call involves

polynomial-time computations; evaluating Hasse derivatives, performing polynomial divisions, verifying agreement, and executing $\mathrm{poly}(n)$ many calls to the Newton Iteration procedure (Lemma 2.1), each of which requires $\mathrm{poly}(n)$ field operations.

Therefore, the total running time of the algorithm is $\mathrm{poly}(n, \log|\mathbb{F}|)$, and the algorithm is deterministic, completing the proof of Theorem 4.1.

5 Derandomizing the Guruswami-Sudan decoder

In this section, we prove Theorem 1.1. We begin by recalling the high-level structure of the Guruswami–Sudan algorithm [GS99], which proceeds in two stages.

1. **Interpolation.** Given the received word $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$, find a nonzero bivariate polynomial $Q(X, Y)$ of $(1, k-1)$-degree at most D such that each point (α_j, β_j) is a root of Q with multiplicity at least m. To decode from agreement $\sqrt{n(k-1)}$, set $m = 2\sqrt{n(k-1)}$ and $D = \Theta(m\sqrt{(k-1)n})$.

2. **Factorization.** Find all factors of $Q(X, Y)$ of the form $(Y - f(X))$, where $\deg f \leq k-1$ and f agrees with w on at least $\sqrt{(k-1)n}$ coordinates.

The following two lemmas from [GS99] ensure that the interpolation step succeeds and that all sufficiently close codewords appear as Y-roots of Q.

Lemma 5.1 (interpolation guarantees [GS99])**.** *For every $n > k \in \mathbb{N}$, there exist parameters $m = 2\sqrt{n(k-1)}$ and $D = \Theta(m\sqrt{(k-1)n})$ such that the following hold.*

For every received word $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$, there exists a nonzero polynomial $Q(X, Y)$ satisfying:

1. *$(1, k-1)$-degree$(Q) \leq D$,*
2. *$\deg_Y(Q) = O(m\sqrt{n/(k-1)})$,*
3. *For every $j \in [n]$, Q vanishes at each (α_j, β_j) with multiplicity at least m.*

Moreover, such a Q can be found deterministically in $\mathrm{poly}(n, \log|\mathbb{F}|)$ time.

Lemma 5.2 (root containment [GS99])**.** *Let Q satisfy the properties in Lemma 5.1. Then for every polynomial f of degree at most $k-1$ that agrees with w on at least $\sqrt{(k-1)n}$ coordinates, we have $Q(X, f(X)) \equiv 0$, i.e., $(Y - f(X))$ divides $Q(X, Y)$.*

Proof. The proof follows immediately from the bounds on D and m in Lemma 5.1 and the observation that at each agreement point of f and w, the univariate $Q(X, f(X))$ vanishes with multiplicity at least m. □

Thus, the list-decoding task reduces to finding all Y-roots of $Q(X,Y)$ of degree at most $(k-1)$. The main challenge is to do this deterministically in time $\mathrm{poly}(n, \log|\mathbb{F}|)$. The next theorem, which is the main contribution of this paper, provides such an algorithm.

Theorem 5.3. *There is a deterministic algorithm such that for every finite field $\mathbb{F}$ and parameters $n > k \in \mathbb{N}$:*

- *It takes as input a received word $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$ and a polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$ with $\deg_Y(Q) = \mathrm{poly}(n)$, under the promise that every degree-$(k-1)$ polynomial with agreement at least $\sqrt{(k-1)n}$ with w is a Y-root of Q.*
- *It outputs all such polynomials f in time $\mathrm{poly}(n, \log|\mathbb{F}|)$.*

The rest of this section proves this theorem, completing the proof of Theorem 1.1.

Let L denote the set of polynomials $f(X)$ of degree less than k and agreement more than $\sqrt{(k-1)n}$ with w. Factor $Q(X,Y) = \prod_{i=1}^{s} Q_i(X,Y)$, where each Q_i is (with multiplicity) and has $\deg_Y(Q_i) \geq 1$. We may assume without loss of generality that Q has no factors depending only on X, since dividing out the GCD of its pure-X components does not affect the property that each $f \in L$ is a Y-root of Q.

In the rest of this section we prove this theorem, and this would complete the proof of Theorem 1.1. We start by recalling some notation already discussed in the proof overview. Let L be the set of polynomials $f(X)$ of degree less than k and agreement more than $\sqrt{(k-1)n}$ with the received word w.

5.1 Local Splitting

Let $P(X,Y) \in \mathbb{F}[X,Y]$ be a bivariate polynomial with no pure-X factors and a point $(\alpha,\beta) \in \mathbb{F}^2$. Given the factorization $P = \prod_{i=1}^{s} P_i$ into irreducibles (with multiplicity), define:

$$
\begin{aligned}
A(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha, Y) = \gamma\,(Y-\beta)^{m_i} \text{ for some } m_i \geq 1,\ \gamma \in \mathbb{F}\setminus\{0\}\},\\
B(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha,\beta) \neq 0, \deg P_i(\alpha, Y) \geq 1\},\\
C(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha, Y) = (Y-\beta)^{m_i} \cdot \widehat{P}_i(Y) \text{ for some } m_i \geq 1, \deg_Y \widehat{P}_i \geq 1, \widehat{P}_i(\beta) \neq 0\},\\
D(\alpha,\beta) &:= \{i \in [s] \mid P_i(\alpha,\beta) \neq 0, \deg_Y P_i(\alpha, Y) = 0\}.
\end{aligned}
$$

Let $P_{A(\alpha,\beta)} = \prod_{i\in A(\alpha,\beta)} P_i$, and define $P_{B(\alpha,\beta)}$, $P_{C(\alpha,\beta)}$, $P_{D(\alpha,\beta)}$ analogously. For each (α_j,β_j) in w, we write A_j, B_j, C_j, D_j and $P_{A,j}, P_{B,j}, P_{C,j}, P_{D,j}$ for brevity instead of $A(\alpha_j,\beta_j), B(\alpha_j,\beta_j), C(\alpha_j,\beta_j)$, $D(\alpha_j,\beta_j)$ and $P_{A(\alpha_j,\beta_j)}, P_{B(\alpha_j,\beta_j)}, P_{C(\alpha_j,\beta_j)}, P_{D(\alpha_j,\beta_j)}$ for (α_j,β_j).

We recall that in Section 2, we focused on bivariates that are monic in Y. For such polynomials, all the irreducible factors can be assumed to be monic in Y without loss of generality. As a consequence, the Y-degree of any of its irreducible factors cannot drop when X is set to $\alpha \in \mathbb{F}$ and

hence the set D defined above is empty. However, the Guruswami-Sudan interpolation polynomial Q need not be monic in Y and we need to exercise some additional care to handle such polynomials.

We now apply this definition to Guruswami–Sudan interpolation polynomial Q (or more precisely Q with its pure X factors shaved off). If $f \in L$, then $Q(X, f(X)) \equiv 0$ or equivalently $Y - f(X) \mid Q$. Now for any $j \in [n]$ if $f(\alpha_j) = \beta_j$ then $Y - f(X) \equiv Y - \beta_j \bmod \langle X - \alpha_j \rangle$. Hence, $Y - f(X) \mid Q_{A,j}(X, Y)$. And if $f(\alpha_j) \neq \beta_j$ then $(Y - f(X))(\alpha_j, \beta_j) = \beta_j - f(\alpha_j) \neq 0$. Therefore, $Y - f(X) \mid Q_{B,j}(X, Y)$. Therefore, we have the following observation:

Observation 5.4. *For every $f \in L$:*

1. *If $f(\alpha_j) = \beta_j$ for some j, then $(Y - f(X)) \mid Q_{A,j}(X, Y)$.*
2. *If $f(\alpha_j) \neq \beta_j$ for some j, then $(Y - f(X)) \mid Q_{B,j}(X, Y)$.*

We also rely on the following simple observation in the description of our algorithm.

Observation 5.5. *Let $g(X, Y) \in \mathbb{F}[X, Y]$ be a bivariate polynomial such that there is an $\alpha \in \mathbb{F}$ for which $g(\alpha, Y)$ is a non-zero field constant. Then, $g(X, Y)$ is not divisible any bivariate polynomial that is monic in Y.*

In particular, $g(X, Y)$ does not have a factor of the form $(Y - f(X))$.

Proof. We will prove the contrapositive - we assume that $g(X, Y)$ has a factor $P(X, Y)$ that is monic in Y and show that $g(\alpha, Y)$ cannot be a non-zero field constant.

Let $R(X, Y)$ be a polynomial that is monic in Y (and hence has Y-degree at least one) and divides $g(X, Y)$. Thus, there is a bivariate R' such that $g(X, Y) = R(X, Y)R'(X, Y)$. Now, by setting X to α in the above identity, we get $g(\alpha, Y) = R(\alpha, Y)R'(\alpha, Y)$. Since $g(\alpha, Y)$ is non-zero, both $R'(\alpha, Y)$ and $R(\alpha, Y)$ are non-zero polynomials. Moreover, R was monic in Y, so its degree in Y does not reduce when X is set to α. But this implies that the $g(\alpha, Y)$ has degree at least as $R(\alpha, Y)$, which is at least one. □

This observation motivates the following definition.

Definition 5.6 (Useless bivariates)**.** *A bivariate $g(X, Y) \in \mathbb{F}[X, Y]$ is said to be useless with respect to a set $E \subseteq \mathbb{F}$ if there exists an $\alpha \in E$ such that $g(\alpha, Y)$ is a non-zero field constant.* ◊

We now define the notion of stability, that is once again crucially used in our algorithm.

Definition 5.7 (stability of a polynomial)**.** *We call a polynomial $P(X, Y)$* stable *with respect to the point (α, β) if $\deg P(\alpha, Y) \geq 1$ and one of the following holds*

- *$P(\alpha, Y) = \gamma(Y - \beta)^r$ for some $\gamma \in \mathbb{F} \setminus \{0\}$ and positive integer r*
- *$P(\alpha, \beta) \neq 0$*

We call a polynomial $P(X,Y)$ stable *with respect to the received word w if for all $j \in [n]$, P is stable with respect to (α_j, β_j). We will usually drop the phrase "with respect to the received word w"*

A finite set $S \subseteq \mathbb{F}[X,Y]$ is stable if every $P \in S$ is stable. ◊

Our algorithm relies on a deterministic subroutine, Split, that given $P(X,Y)$ and (α,β) computes $P_{A(\alpha,\beta)}R_1$ and $P_{B(\alpha,\beta)}R_2$ such that $R_1 \cdot R_2$ divides $P_{D(\alpha,\beta)}$ in time $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$. We defer its description to Section 6 and restate its guarantee below.

Theorem 5.8 (Split procedure)**.** *Let $\mathbb{F}$ be any finite field. There is a deterministic algorithm Split which, given $P \in \mathbb{F}[X,Y]$ and $(\alpha,\beta) \in \mathbb{F}^2$, outputs polynomials (P_1, P_2) such that $P_1 = P_{A(\alpha,\beta)} \cdot R_1$ and $P_2 = P_{B(\alpha,\beta)} \cdot R_2$ where $R_1 \cdot R_2$ divides $P_{D(\alpha,\beta)}$.*

Moreover, if P is stable with respect to (α,β), Split outputs either $(P,1)$ or $(1,P)$ depending on which of the two conditions in Definition 5.7 is met.

The *moreover* part of the above theorem explains the behavior of the algorithm Split on inputs that are stable. The corollary below summarises its properties on inputs that are not stable. This will be useful for us in the analysis of the algorithm later in this section.

Corollary 5.9. *Let $P(X,Y)$ be a polynomial such that the degree of $P(\alpha,Y)$ is at least one and P is not stable at the point (α,β). Let (P_1,P_2) be the output of the algorithm Split on inputs $P,(\alpha,\beta)$.*

Then, $\deg_Y(P_1), \deg_Y(P_2)$ are both strictly smaller than $\deg_Y(P)$. Moreover, $\deg_Y(P_1)+\deg_Y(P_2) \leq \deg_Y(P)$.

Proof. The *moreover* part of the corrollary follows immediately from the structure and guarantees on P_1, P_2 in Theorem 5.8. So, we focus on proving that the Y-degrees of both P_1 and P_2 is strictly smaller than the Y-degree of P.

Since $\deg_Y(P(\alpha,Y))$ is at least one, and P is not stable at (α,β), we get from Definition 5.7 that $P(\alpha,\beta)$ is zero and $P(\alpha,Y)$ is not a non-zero scalar multiple of a power of $(Y-\beta)$. In other words, there is a univariate $\widehat{P} \in \mathbb{F}[Y]$ of degree at least one, a positive integer r such that

$$P(\alpha,Y) = (Y-\beta)^r \cdot \widehat{P}(Y),$$

and $\widehat{P}(\beta)$ is non-zero.

From the above and the definitions of $P_{A(\alpha,\beta)}, P_{B(\alpha,\beta)}, P_{C(\alpha,\beta)}, P_{D(\alpha,\beta)}$, we get that the product $P_{B(\alpha,\beta)}(\alpha,Y) \cdot P_{C(\alpha,\beta)}(\alpha,Y)$ must be divisible by $\widehat{P}$, and hence has degree at least one. Furthermore, from Theorem 5.8, we have that P_1 divides $P_{A(\alpha,\beta)}P_{D(\alpha,\beta)}$. Hence, we have the following sequence

of inequalities.

$$\begin{aligned}
\deg_Y(P_1) &\le \deg_Y(P_{A(\alpha,\beta)}(X,Y)) + \deg_Y(P_{D(\alpha,\beta)}(X,Y)) \\
&= \deg_Y(P) - (\deg_Y(P_{B(\alpha,\beta)}(X,Y)) + \deg_Y(P_{C(\alpha,\beta)}(X,Y))) \\
&\le \deg_Y(P) - (\deg(P_{B(\alpha,\beta)}(\alpha,Y)) + \deg(P_{C(\alpha,\beta)}(\alpha,Y))) \\
&\le \deg_Y(P) - 1.
\end{aligned}$$

Here, we used the fact that the Y degree cannot increase when X is set to α and the fact that $(\deg(P_{B(\alpha,\beta)}(\alpha,Y)) + \deg(P_{C(\alpha,\beta)}(\alpha,Y)))$ is equal to the degree of the product $P_{B(\alpha,\beta)}(\alpha,Y) \cdot P_{C(\alpha,\beta)}(\alpha,Y)$, which as discussed earlier is at least one.

Similarly, since r is positive, we get that the product $P_{A(\alpha,\beta)}(\alpha,Y) \cdot P_{C(\alpha,\beta)}(\alpha,Y)$ must be divisible by $(Y-\beta)^r$, and hence has degree at least one. We again have a sequence of inequalities.

$$\begin{aligned}
\deg_Y(P_2) &\le \deg_Y(P_{B(\alpha,\beta)}(X,Y)) + \deg_Y(P_{D(\alpha,\beta)}(X,Y)) \\
&= \deg_Y(P) - (\deg_Y(P_{A(\alpha,\beta)}(X,Y)) + \deg_Y(P_{C(\alpha,\beta)}(X,Y))) \\
&\le \deg_Y(P) - (\deg(P_{A(\alpha,\beta)}(\alpha,Y)) + \deg(P_{C(\alpha,\beta)}(\alpha,Y))) \\
&\le \deg_Y(P) - 1.
\end{aligned}$$

□

5.2 Details of the algorithm

We are now ready to describe the deterministic variant of the Guruswami–Sudan algorithm. Given $Q(X,Y)$, we first remove any pure-X factors to obtain Q'. The algorithm then proceeds in two stages:

1. **Refinement.** Initialize $S \leftarrow \{Q'\}$. Repeatedly apply SPLIT to every $P \in S$ and each (α_j, β_j), replacing P by its components $P_{A,j}, P_{B,j}$, until S becomes stable.
2. **Extraction.** For each $g \in S$, let $U = \{j : g(\alpha_j, \beta_j) = 0\}$. If there exists a degree-$(k-1)$ polynomial f that agrees with w on all indices in U, output f.

The formal description of the algorithm follows.

Algorithm 2: DETERMINISTIC-FACTORIZATION-FOR-LIST-DECODING

Input: Received word $w = \{(\alpha_j, \beta_j)\}_{j\in[n]}$ and interpolation polynomial $Q(X, Y) \in \mathbb{F}[X, Y]$ such that $Q(\alpha_j, \beta_j) = 0$ for all $j \in [n]$.

Output: All polynomials $f \in \mathbb{F}[X]$ of degree at most $(k-1)$ that agree with w on at least $\sqrt{(k-1)n}$ coordinates.

Step 1: Remove pure-X factors.

Write $Q(X, Y) = \sum_\ell Q^{(\ell)}(X)Y^\ell$ and set $\tilde{Q} \leftarrow \gcd(Q^{(1)}, Q^{(2)}, \ldots)$.

Set $Q' \leftarrow Q/\tilde{Q}$ and initialize $S \leftarrow \{Q'(X, Y)\}$.

if Q' *is* useless *with respect to* $\{\alpha_1, \ldots, \alpha_n\}$ *as per Definition 5.6* **then**

  return $\emptyset$

Step 2: Refinement via SPLIT.

while *there is a* $g \in S$ *of* Y*-degree at least* 2 *that is not stable* **do**

  Choose a $j \in [n]$ such that g is not stable with respect to (α_j, β_j)

  Update $S \leftarrow S \setminus \{g\}$.

  Set $(g_1, g_2) \leftarrow \text{SPLIT}(g, (\alpha_j, \beta_j))$

  if $\deg_Y(g_1) \geq 1$ **then**

    Write $g_1(X, Y) = \sum_\ell g_1^{(\ell)}(X)Y^\ell$ and set $\tilde{g}_1 \leftarrow \gcd(g_1^{(1)}, g_1^{(2)}, \ldots)$.

    if $g_1/\tilde{g}_1$ *is not* useless *with respect to* $\{\alpha_1, \ldots, \alpha_n\}$ *as per Definition 5.6* **then**

      $S \leftarrow S \cup \{g_1/\tilde{g}_1\}$

  if $\deg_Y(g_2) \geq 1$ **then**

    Write $g_2(X, Y) = \sum_\ell g_2^{(\ell)}(X)Y^\ell$ and set $\tilde{g}_2 \leftarrow \gcd(g_2^{(1)}, g_2^{(2)}, \ldots)$.

    if $g_2/\tilde{g}_2$ *is not* useless *with respect to* $\{\alpha_1, \ldots, \alpha_n\}$ *as per Definition 5.6* **then**

      $S \leftarrow S \cup \{g_2/\tilde{g}_2\}$

Step 3: Extract codewords.

Initialize $L \leftarrow \emptyset$.

for $g \in S$ **do**

  if $\deg(g) = 1$ *and* g *is of the form* $\gamma(Y - f)$ *for* $\gamma \cdot \in \mathbb{F} \setminus \{0\}$, $deg(f) < k$ **then**

    if f *agrees with* w *on at least* $\sqrt{(k-1)n}$ *coordinates* **then** $L \leftarrow L \cup \{f\}$.

  else

    Set $U \leftarrow \{j \in [n] : g(\alpha_j, \beta_j) = 0\}$.

    if *there exists* $f \in \mathbb{F}[X]$ *with* $\deg(f) \leq (k-1)$ *and* $f(\alpha_j) = \beta_j$ *for all* $j \in U$ **then**

      if f *agrees with* w *on at least* $\sqrt{(k-1)n}$ *coordinates* **then** $L \leftarrow L \cup \{f\}$.

return L

5.3 Correctness of the algorithm

We now argue the correctness of Algorithm 2.

If the polynomial Q' is useless with respect to $\{\alpha_1, \ldots, \alpha_n\}$, then, from Observation 5.5, we get that Q' has no factors of the form $Y - f$ for an $f \in \mathbb{F}[X]$ and the algorithm is clearly correct and terminates in polynomial time. Thus, for the rest of the this discussion, we assume that this is not the case.

Suppose the while loop terminates after T iterations, and let S_t denote the set of polynomials at the end of the t-th iteration, for $t \in \{0, \ldots, T\}$. Thus $S_0 = \{Q'(X, Y)\}$ and S_T is the final stable set.

To prove correctness, we proceed in two steps:

1. For every f in the true list L and every iteration t, there exists some $g \in S_t$ such that $(Y - f(X)) \mid g(X, Y)$.

2. For the final stable set S_T, the zeros of such a g correspond exactly to the agreement points of f with the received word w.

Since the number of agreement points exceeds $\sqrt{(k-1)n} > k-1$, the polynomial f can be uniquely recovered by interpolation in Step 3 of Algorithm 2.

We now formalize these claims, but before that state the following simple observation that is immediately evident from the description of the algorithm.

Observation 5.10. *For every $t \in \{1, 2, \ldots, T\}$, and for every polynomial $g(X, Y)$ in the set S_t, the following are true.*

- *g is not useless with respect to $\{\alpha_1, \ldots, \alpha_n\}$.*
- *If $g = \sum_\ell g_i(X) Y^\ell$, then $\gcd(g_0(X), g_1(X), \ldots)$ equals 1. In particular, g has no pure X factors.*

We now state and prove the first technical lemma.

Lemma 5.11. *For every $f \in L$ and every $t \in \{0, \ldots, T\}$, there exists a polynomial $g \in S_t$ such that $(Y - f(X)) \mid g(X, Y)$.*

Proof. We proceed by induction on t. For the base case $t = 0$, the claim follows directly from Lemma 5.2, since $(Y - f(X)) \mid Q(X, Y)$ (and hence, $(Y - f) \mid Q'$) for every $f \in L$.

Assume the claim holds for S_{t-1}, and let $g \in S_{t-1}$ be such that $(Y - f(X)) \mid g(X, Y)$. Consider the t-th iteration of the while loop, when g is selected for splitting at some point (α_j, β_j). For this to happen the Y-degree of g is at least two.

If g is already stable, it remains in S_t, and the inductive claim holds trivially. Otherwise, $(g_1, g_2) \leftarrow \textsc{Split}(g, (\alpha_j, \beta_j))$. By Observation 5.4, if $f(\alpha_j) = \beta_j$, then $(Y - f(X))$ divides $g_{A,j}$;

otherwise it divides $g_{B,j}$. From Theorem 5.8, we get that in the first case, $(Y - f(X))$ divides g_1 and in the latter case, it divides g_2.

If $Y - f$ divides g_1, then g_1 clearly has Y-degree at least one, and from Observation 5.5, it cannot be useless with respect to $\{\alpha_1, \ldots, \alpha_n\}$, and it is added to the set S_t (after removing some pure X factors). Similarly, if $Y - f$ divides g_2, then g_2 clearly has Y-degree at least one, and from Observation 5.5, it cannot be useless with respect to $\{\alpha_1, \ldots, \alpha_n\}$, and it is added to the set S_t (again after removing some pure X factors).

Thus S_t continues to contain a polynomial divisible by $(Y - f(X))$, completing the induction. □

Lemma 5.12. *For every $f \in L$, there exists $g \in S_T$ such that, for all $j \in [n]$,*

$$g(\alpha_j, \beta_j) = 0 \iff f(\alpha_j) = \beta_j.$$

Proof. From Lemma 5.11, there exists $g \in S_T$ with $(Y - f(X)) \mid g(X, Y)$. We work with this g for the rest of the proof.

Clearly, $g(\alpha_j, \beta_j) = 0$ whenever $f(\alpha_j) = \beta_j$. Suppose for contradiction that there exists j such that $g(\alpha_j, \beta_j) = 0$ but $f(\alpha_j) \neq \beta_j$. Then g must have some irreducible factor $\widehat{g}(X, Y)$ distinct from $(Y - f(X))$ with $\widehat{g}(\alpha_j, \beta_j) = 0$ and Y-degree at least 1 (the latter is true because we have shaved off all pure X factors). Furthermore, $\widehat{g}(\alpha_j, Y)$ also has Y-degree at least one, since if this is not the case, the only way $\widehat{g}(\alpha_j, \beta_j)$ can be zero is if $\widehat{g}(\alpha_j, Y)$ is zero. But this would mean that g has pure X-factors, hence contradicting Observation 5.10.

Thus, the univariate $g(\alpha_j, Y)$ has Y-degree at least two, and is divisible by $(Y - f(\alpha_j))$ and $\widehat{g}(\alpha_j, Y)$, both of which have Y degree at least one. From Definition 5.7, we get that g is not stable at (α_j, β_j), contradicting the stability of S_T.

Therefore, the zeros of $g(X, Y)$ coincide exactly with the agreement points of f and w. □

By Lemma 5.12, for every $f \in L$ there exists a polynomial $g \in S_T$ whose zero set identifies precisely the coordinates where f and w agree. Since the number of agreement points is greater than $\sqrt{(k-1)n}$ and thus exceeds $k - 1$, f can be uniquely recovered by interpolation in Step 3 of Algorithm 2.

This completes the proof of correctness of the deterministic list-decoding algorithm.

5.4 Running-time analysis

The algorithm is clearly deterministic (assuming Theorem 5.8 for the Split subroutine). We now show that it runs in time $\text{poly}(n, \log |\mathbb{F}|)$. The extraction step (Step 3) of Algorithm 2 runs in $\text{poly}(n, \log |\mathbb{F}|, |S|)$ time, where S is the set of polynomials obtained after refinement (Step 2). Hence, it suffices to show that the refinement step (Step 2) terminates in polynomial time and that $|S|$ is polynomially bounded.

Lemma 5.13. *The refinement step (Step 2) of Algorithm 2 terminates in* $\mathrm{poly}(\deg(Q), \log|\mathbb{F}|) = \mathrm{poly}(n, \log|\mathbb{F}|)$ *time.*

Proof. Let S denote the set of polynomials at any stage of refinement step (Step 2). We now show, via a potential-function argument, that the refinement process terminates after only polynomially many updates.

Partition S by Y-degree: let $S^{(i)} = \{g \in S \mid \deg_Y(g) = i\}$ for $i \in [\deg_Y(Q)]$. Define the potential

$$\Phi(S) = \sum_{i=1}^{\deg_Y(Q)} (i-1)\,|S^{(i)}|.$$

Initially, $S = \{Q(X,Y)\}$ and $\Phi(S) = \deg_Y(Q) - 1$. We will show that if S is not already stable, then $\Phi(S)$ decreases by at least 1 in the next iteration, implying that the refinement loop executes at most $\deg_Y(Q)$ times.

Consider a refinement step where the set S is modified to the set S' by splitting $g \in S$ with respect to the point (α_j, β_j). For g to be split, the Y-degree of g must be at least two and there must be a point (α_j, β_j) such that g is not stable at (α_j, β_j). Let (g_1, g_2) be the output of Split on input $g, (\alpha_j, \beta_j)$.

Note that if neither of the polynomials $g_1' = g_1/\tilde{g_1}$ and $g_2' = g_2/\tilde{g_2}$ is added to the set S (to obtain S') in this iteration, then the potential for S' is clearly strictly less than that of S, since the polynomial g with Y-degree at least two is removed from S in the iteration. Thus, at least one of g_1', g_2' is in S'. We now argue that the potential of S' is strictly smaller than the potential of S using some case analysis.

Let $d = \deg_Y(g)$, $d_1 = \deg_Y(g_1')$ and $d_2 = \deg_Y(g_2')$. We now consider some cases.

- $d_1 \geq 1, d_2 \geq 1$**:** In this case, $\Phi(S') - \Phi(S) \leq -(d-1) + (d_1 - 1) + (d_2 - 1)$, which is clearly at most -1 since $d_1 + d_2 \leq d$ from Corollary 5.9.
- $d_1 = d_2 = 0$**:** In this case, $\Phi(S') - \Phi(S) = -(d-1)$ which is at most -1 since d is at least 2 (else g could not have been chosen for splitting).
- $d_1 \geq 1, d_2 = 0$**:** In this case, $\Phi(S') - \Phi(S) \leq -(d-1) + (d_1 - 1) = d_1 - d$, but this is again at most -1 since $d_1 < d$ from Corollary 5.9.
- $d_2 \geq 1, d_1 = 0$**:** This is the same as the previous case.

Since the initial potential is at most $\deg_Y(Q) - 1$, the refinement loop can execute at most $\deg_Y(Q)$ iterations, each of which runs in $\mathrm{poly}(\deg(Q), \log|\mathbb{F}|)$ time. Hence, Step 2 terminates in $\mathrm{poly}(n, \log|\mathbb{F}|)$ time. □

6 The Split algorithm

In this section, we prove Theorem 5.8, completing the proof of Theorem 5.3 and hence of Theorem 1.1. For completeness, we restate the theorem below.

Theorem 5.8 (Split procedure)**.** *Let $\mathbb{F}$ be any finite field. There is a deterministic algorithm Split which, given $P \in \mathbb{F}[X,Y]$ and $(\alpha,\beta) \in \mathbb{F}^2$, outputs polynomials (P_1, P_2) such that $P_1 = P_{A(\alpha,\beta)} \cdot R_1$ and $P_2 = P_{B(\alpha,\beta)} \cdot R_2$ where $R_1 \cdot R_2$ divides $P_{D(\alpha,\beta)}$.*

Moreover, if P is stable with respect to (α,β), Split outputs either $(P,1)$ or $(1,P)$ depending on which of the two conditions in Definition 5.7 is met.

6.1 The Hensel lifting paradigm

Our main ingredient is the classical technique of *Hensel lifting*, which allows one to lift a local factorization of a polynomial modulo a power of $(X-\alpha)$ to a higher power. This may be viewed as the analog of Newton iteration for factoring, as opposed to root lifting. We recall the version we use below and refer to [ST21, KSS15] for a proof.

Lemma 6.1 (Hensel Lifting [ST21])**.** *Let $\mathbb{F}$ be a finite field, and let $P, g, h, a, b \in \mathbb{F}[X,Y]$, $\alpha \in \mathbb{F}$, and $m \geq 1$ satisfy*

$$P \equiv gh \bmod \langle X-\alpha\rangle^m \quad \text{and} \quad ag + bh \equiv 1 \bmod \langle X-\alpha\rangle^m.$$

Then the following hold.

1. *(Existence) There exist $g', h' \in \mathbb{F}[X,Y]$ such that*
   - *(a) $P \equiv g'h' \bmod \langle X-\alpha\rangle^{2m}$*
   - *(b) $g' \equiv g \bmod \langle X-\alpha\rangle^m$ and $h' \equiv h \bmod \langle X-\alpha\rangle^m$*
   - *(c) there exist $a', b' \in \mathbb{F}[X,Y]$ with $a'g' + b'h' \equiv 1 \bmod \langle X-\alpha\rangle^{2m}$.*

   We call g', h' the lifts *of g, h respectively and a', b' the corresponding* Bezout pairs.
2. *(Uniqueness) For any other lifts g^*, h^* of g, h, there exists $u \in \langle X-\alpha\rangle^m$ such that*

$$g^* \equiv g'(1+u) \bmod \langle X-\alpha\rangle^{2m}, \qquad h^* \equiv h'(1-u) \bmod \langle X-\alpha\rangle^{2m}.$$

3. *(Efficiency) There is a deterministic algorithm that, given P, g, h, a, b, m, α, outputs g', h', a', b' in time $\mathrm{poly}(\deg(P), \deg(a), \deg(b), \log|\mathbb{F}|, m)$.*

Degree bounds under iterative Hensel lifting. In standard applications of Hensel lifting for polynomial factorization, the polynomial P is typically assumed to be monic, in which case degree growth during lifting is straightforward to bound. In our setting, we must handle the non-monic case as well. The following lemma provides quantitative degree bounds for the iterative lifting process; these follow from the analysis in [Sud12, Lecture 10].

Lemma 6.2 (Degree bounds for iterative Hensel lifting)**.** *Let $\mathbb{F}$ be any field, and let $P(X, Y)$ be a polynomial of total degree d. Fix $\alpha \in \mathbb{F}$.*

1. ***(Degree bound)*** *Let g_0, h_0 be relatively prime polynomials of positive Y-degree such that*

$$P \equiv g_0 h_0 \bmod \langle X - \alpha \rangle.$$

Let (g_i, h_i) and (a_i, b_i) be the sequences of lifts and Bézout pairs obtained by successive applications of Lemma 6.1, satisfying

$$\begin{aligned} &P \equiv g_i h_i \bmod \langle X - \alpha \rangle^{2^i}, \\ &a_i g_i + b_i h_i \equiv 1 \bmod \langle X - \alpha \rangle^{2^i}, \\ &g_i \equiv g_{i-1} \bmod \langle X - \alpha \rangle^{2^{i-1}}, \quad h_i \equiv h_{i-1} \bmod \langle X - \alpha \rangle^{2^{i-1}}. \end{aligned}$$

Then, for every i, the total degrees of a_i, b_i, g_i, h_i are at most $d \cdot 5^i$.

2. ***(Uniqueness)*** *Let (g_i', h_i') be another sequence of pairs of total degree at most D satisfying the same congruences as above and with $g_1' \equiv g_0$, $h_1' \equiv h_0$ modulo $\langle X - \alpha \rangle$. Then there exist polynomials $u_0, u_1, \ldots$ such that $u_i \in \langle X - \alpha \rangle^{2^i}$ and has total degree at most $(d + D)10^i$ satisfying*

$$g_i' \equiv g_i \cdot \left(\prod_{j<i} (1 + u_j) \right) \bmod \langle X - \alpha \rangle^{2^i}, \quad h_i' \equiv h_i \cdot \left(\prod_{j<i} (1 + u_j) \right)^{-1} \bmod \langle X - \alpha \rangle^{2^i}.$$

Proof. We closely follow the proof of Lemma 6.1 from [ST21].

Degree Bound: From the proof of Lemma 6.1 for any i we have the following relations:

1. $m_i = f - g_i \cdot h_i$;
2. $g_{i+1} = g + b_i \cdot m_i, \quad h_{i+1} = g_i + a_i \cdot m_i$;
3. $q_{i+1} = a_i \cdot g_{i+1} + b_i \cdot h_{i+1} - 1$;
4. $a_{i+1} = a_i - a_i \cdot q_{i+1}, \quad b_{i+1} = b_i - b_i \cdot q_{i+1}$.

For $i = 0$, we have $\deg(g_0), \deg(h_0) \leq d$, since $\deg(P(\alpha, Y)) \leq \deg(P) \leq d$. As $g_0, h_0 \in \mathbb{F}[Y]$, the extended GCD algorithm ensures that $\deg(a_0) < \deg(h_0) \leq d$ and $\deg(b_0) < \deg(g_0) \leq d$. Hence, the base case holds.

Assume inductively that the total degrees of (g_i, h_i, a_i, b_i) are at most $d \cdot 5^i$. Then $\deg(g_i \cdot h_i) \leq 2d \cdot 5^i$ and hence $\deg(m_i) \leq 2d \cdot 5^i$. Consequently $\deg(g_{i+1}) \leq \deg(b_i \cdot m_i) \leq 3d \cdot 5^i$ and similarly, $\deg(h_{i+1}) \leq 3d \cdot 5^i$. So, $\deg(a_i \cdot g_{i+1}) \leq 4d \cdot 5^i$ and similarly, we have $\deg(b_i \cdot h_{i+1}) \leq 4d \cdot 5^i$. Hence, $\deg(q_{i+1}) \leq 4d \cdot 5^i$. Thus, $\deg(a_{i+1}) \leq \deg(a_i \cdot q_{i+1}) \leq d \cdot 5^i + 4d \cdot 5^i = d \cdot 5^{i+1}$ and similalry, $\deg(b_{i+1}) \leq d \cdot 5^{i+1}$. So we got the degree bounds of $(g_{i+1}, h_{i+1}, a_{i+1}, b_{i+1})$ are at most $d \cdot 5^{i+1}$. By induction, we conclude that for every $i \geq 0$, the total degrees of g_i, h_i, a_i, b_i are bounded by $d \cdot 5^i$.

Uniqueness: We now establish uniqueness by induction on i. For the base case $i = 1$, we have $g_1 \equiv g_1' \equiv g_0 \bmod \langle X - \alpha \rangle$ and $h_1 \equiv h_1' \equiv h_0 \bmod \langle X - \alpha \rangle$. Following the proof of Lemma 6.1 define $u_0 = a_1(g_1' - g_1) - b_1(h_1' - h_1)$, so that $u_0 \in \bmod \langle X - \alpha \rangle$ and

$$g_1' \equiv g_1(1 + u_0) \bmod \langle X - \alpha \rangle^2, \quad h_1' \equiv h_1(1 - u_0) \equiv h_1(1 + u_0)^{-1} \bmod \langle X \rangle^2$$

From the degree bounds above, the total degrees of (g_1, h_1, a_1, b_1) are at most $5d$, implying $\deg(a_1(g_1' - g_1)), \deg(b_1(g_1' - g_1)) \leq (d + D)10$ and hence $\deg(u_0) \leq (d + D)10$. Hence, the base case is satisfied.

Now suppose that this is true for i. Since $u_j \in \langle X - \alpha \rangle^{2^j}$ for each $j < i$, it follows that $(1 + u_j)$ is invertible modulo $\langle X - \alpha \rangle^{2^i}$. Now consider the elements $g_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)^{-1}$ and $h_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)$. We have $g_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)^{-1} \equiv g_i' \left(\prod_{j<i}(1 + u_j) \right)^{-1} \equiv g_i \bmod \langle X - \alpha \rangle^{2^i}$ and $h_{i+1}' \left(\prod_{j<i}(1 + u_j) \right) \equiv h_i' \left(\prod_{j<i}(1 + u_j) \right) \equiv h_i \bmod \langle X - \alpha \rangle^{2^i}$. We also have

$$g_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)^{-1} \cdot h_{i+1}' \left(\prod_{j<i}(1 + u_j) \right) \equiv g_{i+1}' h_{i+1}' \equiv f \bmod \langle X - \alpha \rangle^{2^{i+1}}$$

Thus, both g_{i+1}, h_{i+1} and $g_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)^{-1}, h_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)$ are valid lifts of (g_i, h_i). By the Uniqueness property of Lemma 6.1, there exists $u_i \in \langle X - \alpha \rangle^{2^i}$ such that

$$g_{i+1}' \left(\prod_{j<i}(1 + u_j) \right)^{-1} \equiv g_{i+1}(1 + u_i) \bmod \langle X - \alpha \rangle^{2^{i+1}}$$

and therefore $g_{i+1}' \equiv g_{i+1} \prod_{j<i+1}(1 + u_j) \bmod \langle X - \alpha \rangle^{2^{i+1}}$. Similarly,

$$h_{i+1}' \left(\prod_{j<i}(1 + u_j) \right) \equiv h_{i+1}(1 - u_i) \bmod \langle X - \alpha \rangle^{2^{i+1}}$$

which implies $h'_{i+1} \equiv h_{i+1}\left(\prod_{j<i+1}(1+u_j)\right)^{-1} \bmod \langle X-\alpha\rangle^{2^{i+1}}$. Since $(1+u_i)(1-u_i) \equiv 1 \bmod \langle X-\alpha\rangle^{2^{i+1}}$. From the construction in Lemma 6.1, we have $u_i = a_{i+1}(g'_{i+1}-g_{i+1}) - b_{i+1}(h'_{i+1}-h_{i+1})$ and, by the degree bound in the first part, we have the total degrees of $(g_{i+1}, h_{i+1}, a_{i+1}, b_{i+1})$ are at most $d \cdot 5^{i+1}$. Therefore, $\deg(u_i) \le d \cdot 5^{i+1} + \max\{d \cdot 5^{i+1}, D\} \le (d+D)10^{i+1}$. Hence, by induction, for all i, $g'_i \equiv g_i\left(\prod_{j<i}(1+u_j)\right) \bmod \langle X-\alpha\rangle^{2^i}$ and $h'_i \equiv h_i \cdot \left(\prod_{j<i}(1-u_j)\right)^{-1} \bmod \langle X-\alpha\rangle^{2^i}$ with $\deg(u_i) \le (d+D)\cdot 10^i$, as desired. □

6.2 Details of the Split algorithm

We now describe the algorithm promised in Theorem 5.8. For clarity, from now on we write P_A, P_B, P_C, P_D for $P_{A(\alpha,\beta)}, P_{B(\alpha,\beta)}, P_{C(\alpha,\beta)}, P_{D(\alpha,\beta)}$, respectively.

Before proceeding, we record a basic but useful subroutine that we will invoke repeatedly. Suppose the polynomial $P \in \mathbb{F}[X,Y]$ has a nontrivial factor $g \in \mathbb{F}[X,Y]$. If we already know the splits of g and of P/g with respect to a point (α,β), then we can obtain the split of P itself by simply combining the corresponding A and B parts. The following routine formalizes this observation.

Algorithm 3: Combine-Splits

Input: Polynomials $P, g \in \mathbb{F}[X,Y]$ and $(\alpha,\beta) \in \mathbb{F}^2$ with the guarantee that g divides P
Output: Polynomials $P_{A(\alpha,\beta)} \cdot R_1$ and $P_{B(\alpha,\beta)} \cdot R_2$ for some polynomials R_1, R_2 such that $R_1 \cdot R_2$ divide $P_{D(\alpha,\beta)}$

1 Set $g' \leftarrow P/g$
2 Compute $(g_1, g_2) \leftarrow \textsc{Split}(g, (\alpha,\beta))$ and $(g'_1, g'_2) \leftarrow \textsc{Split}(g', (\alpha,\beta))$
3 **return** $(g_1 \cdot g'_1,\ g_2 \cdot g'_2)$

We now develop a series of claims that naturally lead to the Split algorithm. We begin with a deterministic base case that seeds the Hensel lifting.

Lemma 6.3. *Let $\mathbb{F}$ be any finite field. There is a deterministic algorithm that, given a bivariate polynomial $P(X,Y) \in \mathbb{F}[X,Y]$ and a point $(\alpha,\beta) \in \mathbb{F}^2$, runs in time $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$ and outputs a factorization*

$$P(\alpha, Y) = (Y-\beta)^m \cdot \widehat{P}(Y),$$

where $\widehat{P}(\beta) \neq 0$.

Proof. Iteratively perform univariate division to find the largest $m \in \mathbb{N}$ such that $(Y-\beta)^m$ divides $P(\alpha, Y)$; then set $\widehat{P}(Y) := P(\alpha,Y)/(Y-\beta)^m$. Both steps are deterministic and run in time $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$ by standard univariate algorithms. □

An immediate consequence is that if $\deg_Y(P(\alpha, Y)) \geq 1$, we can recognize when $P = P_A P_D$ or $P = P_B P_D$: if $m = \deg_Y(P)$, then $P = P_A P_D$; if $m = 0$, then $P = P_B P_D$. In either case, Theorem 5.8 follows directly. If $\deg_Y(P(\alpha, Y)) = 0$ and $P(\alpha, \beta) \neq 0$, $P = P_D$. In that case we just throw it away.

The interesting case is when $0 < m < \deg_Y(P)$. Then Lemma 6.3 yields

$$P(\alpha, Y) = (Y - \beta)^m \cdot \widehat{P}(Y),$$

where $(Y - \beta)^m$ and $\widehat{P}(Y)$ are coprime and both have positive Y-degree. Equivalently,

$$P(X, Y) \equiv (Y - \beta)^m \cdot \widehat{P}(Y) \pmod{\langle X - \alpha \rangle}.$$

This is precisely the hypothesis needed to invoke Lemma 6.1: we lift this factorization to sufficiently high precision in $(X - \alpha)$, recover a nontrivial factor of P, and then recurse, using Combine-Splits to combine results.

More precisely, set $g_0 = (Y - \beta)^m$ and $h_0 = \widehat{P}(Y)$. Apply iterative Hensel lifting (Lemma 6.1) for $t := 2\lceil \log \deg_Y(P) \rceil + 1$ rounds to obtain g_t, h_t with

$$P(X, Y) \equiv g_t \cdot h_t \bmod \langle X - \alpha \rangle^{2^t}, \qquad g_t \equiv g_0 \bmod \langle X - \alpha \rangle, \quad h_t \equiv h_0 \bmod \langle X - \alpha \rangle.$$

Given these lifts g_t, h_t, we can write the following linear system

$$\begin{aligned} U \cdot h_t &\equiv V \bmod \langle X - \alpha \rangle^{2^t}, \\ E \cdot g_t &\equiv F \bmod \langle X - \alpha \rangle^{2^t}, \end{aligned}$$

where the variables are the coefficients of the polynomials U, V, E, F. The following claim is standard in Hensel-based factorization and we include it for completeness.

Claim 6.4. *If the system $U \cdot h_t \equiv V \bmod \langle X - \alpha \rangle^{2^t}$ has a solution, then $\gcd(V, P)$ has Y-degree at least one. Similarly, if the system $E \cdot g_t \equiv F \bmod \langle X - \alpha \rangle^{2^t}$ has a solution, then $\gcd(F, P)$ has Y-degree at least one.*

Finally, we show that if both systems have no solution, then P must be entirely in the C D-parts.

Claim 6.5. *Let $P(X, Y)$ and (α, β) be inputs with $P \neq P_A P_D$ and $P \neq P_B P_D$. If neither*

$$U \cdot h_t \equiv V \bmod \langle X - \alpha \rangle^{2^t} \quad \textit{nor} \quad E \cdot g_t \equiv F \bmod \langle X - \alpha \rangle^{2^t}$$

has a solution, then $P = P_C P_D$.

Given all the above observations and claims, we now have the following deterministic Split algorithm.

Algorithm 4: SPLIT

Input: $P(X,Y) \in \mathbb{F}[X,Y]$ and a point $(\alpha,\beta) \in \mathbb{F}^2$

Output: Polynomials $P_{A(\alpha,\beta)} \cdot R_1$ and $P_{B(\alpha,\beta)} \cdot R_2$, for some polynomials R_1, R_2 such that $R_1 \cdot R_2$ divide $P_{D(\alpha,\beta)}$.

Step 0: Remove pure-X factors.

Write $P(X,Y) = \sum_\ell P^{(\ell)}(X)\, Y^\ell$ and set $P \leftarrow P/\gcd(P^{(1)}, P^{(2)}, \ldots)$.

Step 1: Seed Hensel lifting and handle simple cases.

Factor $P(\alpha,Y) = (Y-\beta)^m \cdot \widehat{P}(Y)$ such that $\widehat{P}(\beta) \neq 0$ /* Use Lemma 6.3 */

Set $g_0 := (Y-\beta)^m$, $h_0 := \widehat{P}(Y)$.

if $\deg(P(\alpha,Y)) = 0$ **then return** $(1,1)$ /* $P = P_D$ */

if $\deg(g_0) = \deg P(\alpha,Y)$ **then return** $(P,1)$ /* $P = P_A P_D$ */

if $\deg(h_0) = \deg P(\alpha,Y)$ **then return** $(1,P)$ /* $P = P_B P_D$ */

Step 2: Iterative Hensel lifting.

Set $t \leftarrow 2\lceil \log(\deg_Y P) \rceil + 1$ and $\widehat{D} \leftarrow 2\deg(P) \cdot 10^{3\log\deg(P)}$.

Apply iterative Hensel lifting (Lemma 6.1) for t rounds to compute lifts g_t, h_t satisfying

$$P \equiv g_t \cdot h_t \bmod \langle X-\alpha\rangle^{2^t}, \qquad g_t \equiv g_0 \bmod \langle X-\alpha\rangle, \qquad h_t \equiv h_0 \bmod \langle X-\alpha\rangle.$$

Step 3: Solve linear systems.

if *there exist* $U, V \in \mathbb{F}[X,Y]$ *satisfying* $U \cdot h_t \equiv V \bmod \langle X-\alpha\rangle^{2^t}$ *with the degree constraints* $\deg_Y U \le \widehat{D}$, $\deg_Y V \le \deg_Y P - 1$, $\deg_X U \le 2^t - 1$, $\deg_X V \le \deg_X P$ **then**

  Set $\rho \leftarrow \gcd(P,V)$ /* $\deg_Y \rho \ge 1$ by Claim 6.4 */

  return COMBINE-SPLITS$(P,(\alpha,\beta),\rho)$

if *there exist* $E, F \in \mathbb{F}[X,Y]$ *satisfying* $E \cdot g_t \equiv F \bmod \langle X-\alpha\rangle^{2^t}$ *with the degree constraints* $\deg_Y E \le \widehat{D}$, $\deg_Y F \le \deg_Y P - 1$, $\deg_X E \le 2^t - 1$, $\deg_X F \le \deg_X P$ *such that* **then**

  Set $\sigma \leftarrow \gcd(P,F)$ /* $\deg_Y \sigma \ge 1$ by Claim 6.4 */

  return COMBINE-SPLITS$(P,(\alpha,\beta),\sigma)$

Step 4: All tests fail. /* Use Claim 6.5 */

return $(1,1)$.

6.3 Correctness of the SPLIT algorithm

We now establish the correctness of the SPLIT algorithm. We first prove the main correctness lemma assuming Claims 6.4 and 6.5, and subsequently provide proofs of these claims.

Lemma 6.6 (Correctness of SPLIT algorithm)**.** *For every bivariate polynomial $P(X,Y)$ and point $(\alpha,\beta) \in \mathbb{F}^2$, Algorithm 4 terminates and correctly outputs the decomposition $P_A R_1$ and $P_B R_2$ where $R_1 \cdot R_2$ divides P_D.*

Proof. We proceed by induction on $\deg_Y P$.

Base case. If $\deg_Y P = 1$, then P must equal one of P_A, P_B, P_D, and the algorithm's initial checks clearly return the correct classification.

Inductive step. Assume the algorithm is correct for all polynomials of Y-degree at most $d-1$. Let P have $\deg_Y P = d > 1$.

If $P = P_A P_D$ or $P = P_B P_D$, correctness is immediate. Otherwise, both g_0 and h_0 from Step 1 have positive Y-degree, and the algorithm proceeds to the Hensel lifting step to obtain g_t, h_t and then further proceeds to solving the linear systems.

If either linear system admits a solution, then by Claim 6.4, the resulting GCD has positive Y-degree, producing a smaller subproblem on which the inductive hypothesis applies. If neither system has a solution, then by Claim 6.5, $P = P_C P_D$, and the algorithm correctly returns $(1, 1)$.

We now prove the correctness of the algorithm if P is *stable*. Since P is stable, $\deg(P(\alpha, Y)) \geq 1$. Suppose after invoking Lemma 6.3 we obtain the factorization $P(\alpha, Y) = (Y - \beta)^m \cdot \widehat{P}(Y)$ such that $\widehat{P}(\beta) \neq 0$. Now if $P(\alpha, Y) = \gamma(Y - \beta)^r$ for some $\gamma \in \mathbb{F} \setminus \{0\}$ and positive integer r, then, $r = m$ and $\widehat{P}(Y) = \gamma$. Therefore, $\deg(P(\alpha, Y)) = m$ hence the algorithm returns $(P, 1)$ in Step 1. If $P(\alpha, \beta) \neq 0$, $m = 0$. Therefore $\deg(P(\alpha, Y)) = \deg(\widehat{P})$. Therefore, the algorithm returns $(1, P)$ in Step 1.

This completes the inductive argument, and hence the proof of correctness. □

We now prove Claims 6.4 and 6.5.

Proof of Claim 6.4. Consider the resultant $\Gamma(X) := \mathrm{Res}_Y(P, V)$. We have $\deg \Gamma \leq 2 \deg_X(P)^2$. Since $t = 2\lceil \log(\deg_Y P) \rceil + 1$, we have $2^t > \deg \Gamma$. From the property of resultants, there exist $a, b \in \mathbb{F}[X, Y]$ with

$$a \cdot P + b \cdot V = \Gamma(X).$$

Reducing modulo $\langle X - \alpha \rangle^{2^t}$ and using $P \equiv g_t \cdot h_t \bmod \langle X - \alpha \rangle^{2^t}$ and $V \equiv U \cdot h_t \bmod \langle X - \alpha \rangle^{2^t}$ yields

$$(a \cdot g_t + b \cdot U) \cdot h_t \equiv \Gamma \bmod \langle X - \alpha \rangle^{2^t}.$$

Viewing both sides as polynomials in $(X - \alpha)$ with coefficients in $\mathbb{F}[Y]$, the constant term of h_t equals $h_0(Y)$, which has $\deg_Y \geq 1$. However, the right hand side, namely $\Gamma = \mathrm{Res}_Y(P, V)$ has no dependence on Y. Hence, it must be the case that $ag_t + bU$ is divisible by $(X - \alpha)^{2^t}$, forcing $(X - \alpha)^{2^t} \mid \Gamma$. But $\deg \Gamma < 2^t$, so this implies $\Gamma \equiv 0$, i.e., P and V have a nontrivial common factor in $\mathbb{F}(X)[Y]$ (By Lemma 3.1). By Gauss's lemma (see Lemma 3.2), they have a nontrivial GCD in $\mathbb{F}[X, Y]$ of positive Y-degree. The case of E, g_t, F is analogous. □

Proof of Claim 6.5. For brevity, assume wlog $\alpha = 0$ (so the ideal is $\langle X \rangle$). By assumption, $P \neq P_A P_D$ and $P \neq P_B P_D$. We prove the contrapositive: assume $P \neq P_C P_D$ and show that at least one of the two linear systems has a solution.

Since $P = P_A P_B P_C P_D$ and $P \neq P_C P_D$, at least two of P_A, P_B, P_C have positive Y-degree.

Case I: $\deg_Y P_A \geq 1$ and $\deg_Y(P_B P_C) \geq 1$. Reducing modulo $\langle X \rangle$,

$$P_B(X,Y)P_C(X,Y)P_D(X,Y) \equiv (Y-\beta)^m \widehat{h}_0(Y) \pmod{\langle X \rangle}$$

with $\widehat{h}_0(\beta) \neq 0$ and $m \geq 1$. Here $\widehat{h}_0 = h_0$ (since P_A contributes only powers of $(Y-\beta)$). Let $\widehat{g}_0 = (Y-\beta)^m$. Hensel lifting gives lifts $\widehat{g}_t, \widehat{h}_t$ with

$$P_B P_C P_D \equiv \widehat{g}_t \widehat{h}_t \pmod{\langle X \rangle^{2^t}}, \qquad \widehat{g}_t \equiv \widehat{g}_0,\ \widehat{h}_t \equiv \widehat{h}_0 \pmod{\langle X \rangle}.$$

Hence

$$P \equiv \left(P_A \widehat{g}_t\right) \widehat{h}_t \pmod{\langle X \rangle^{2^t}}.$$

Both this factorization and $P \equiv g_t h_t \pmod{\langle X \rangle^{2^t}}$ lift $P(0,Y) = g_0(Y)h_0(Y)$. Since $\widehat{h}_t \equiv h_0 \pmod{\langle X \rangle}$, comparison modulo $\langle X \rangle$ yields $P_A(0,Y)\,\widehat{g}_0(Y) = g_0(Y)$. Using Lemma 6.2 (degree and uniqueness parts), there exist $u_1, \ldots, u_t$ with controlled total degrees and vanishing modulo the appropriate powers of X such that

$$\widehat{h}_t \equiv h_t \prod_{j<t}(1+u_j) \pmod{\langle X \rangle^{2^t}}.$$

Set

$$U := \widehat{g}_t \left(\prod_{j<t}(1+u_j) \right) \bmod \langle X \rangle^{2^t}, \qquad V := P_B P_C.$$

Then $U \cdot h_t \equiv V \pmod{\langle X \rangle^{2^t}}$, with $\deg_X U \leq 2^t - 1$, $\deg_X V \leq \deg_X P$, and (by the chosen parameter bounds) $\deg_Y U \leq \widehat{D}$, $\deg_Y V < \deg_Y P$. Thus the first linear system has a valid solution.

Case II: $\deg_Y P_B \geq 1$ and $\deg_Y(P_A P_C) \geq 1$. An entirely analogous argument (now producing a solution to $E \cdot g_t \equiv F \pmod{\langle X \rangle^{2^t}}$) yields a solution to the second system.

In either case, at least one system has a solution. Therefore, if neither system has a solution, necessarily $P = P_C P_D$. □

6.4 Running time analysis

The algorithm described above is clearly deterministic. We now analyze its running time.

The computations in Step 0 and Step 1 clearly take at most $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$ time. From Lemma 6.1, each Hensel lifting step, that is, computing the lift from (g_{i-1}, h_{i-1}) to (g_i, h_i), requires $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$ time and this is performed for at most $t = \Theta(\log \deg_Y P)$ rounds. In Step 3, the systems of linear equations each have size polynomial in $\deg(P)$, and we solve $O(\deg(P))$ such systems. Hence this step also runs in $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$ time overall.

We now focus on the recursive structure of the algorithm. The key observation is that at most two recursive calls are made before termination. Each call is made with P replaced by polynomials of the form g and P/g, where g is a nontrivial factor of P whose Y-degree is at least one and strictly smaller than $\deg_Y(P)$. Assuming this, the total running time satisfies the recurrence

$$T(d) \;\le\; \max_{r \in \{1,\ldots,d-1\}} \big\{\, T(r) + T(d-r) + \mathrm{poly}(d, \log|\mathbb{F}|) \,\big\}.$$

This immediately implies $T(d) = \mathrm{poly}(d, \log|\mathbb{F}|)$.

To justify the degree constraint, observe first that it holds for the pairs $(\rho, P/\rho)$ and $(\sigma, P/\sigma)$ arising from Step 3, since $\deg_Y(\rho) \le \deg_Y(V)$ and $\deg_Y(V) < \deg_Y(P)$ by Claim 6.4, and likewise for σ.

Therefore, every recursive call reduces the Y-degree strictly, and each level of recursion incurs only polynomial work. This proves that the total running time of the algorithm is $\mathrm{poly}(\deg(P), \log|\mathbb{F}|)$.

Acknowledgements

We thank Daniel Augot and Romain Lebreton for pointing out the reference [AP00], which proves a derandomization of Sudan's algorithm for Reed-Solomon code (and their algebraic geometric variants). We were unaware of this reference at the time of the initial posting of our work.

Mrinal thanks Swastik Kopparty for many helpful discussions on the questions and themes studied in this paper and much encouragement.

References

[Ale05] Michael Alekhnovich. *Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes*. IEEE Trans. Inform. Theory, 51(7):2257–2265, 2005. (Preliminary version in *43rd FOCS*, 2002). 4

[AP00] Daniel Augot and Lancelot Pecquet. *A Hensel lifting to replace factorization in list-decoding of algebraic-geometric and Reed-Solomon codes*. IEEE Trans. Inform. Theory, 46(7):2605–2614, 2000. 4, 6, 32

[Ber67] Elwyn R. Berlekamp. *Factoring polynomials over finite fields*. Bell System Technical Journal, 46(8):1853–1859, 1967. 3

[Ber70] ———. *Factoring polynomials over large finite fields*. Math. Comput., 24(111):713–735, 1970. 3

[CZ81] David G. Cantor and Hans Zassenhaus. *A new algorithm for factoring polynomials over finite fields*. Math. Comput., 36(154):587–592, 1981. 3

[Eli57] Peter Elias. *List decoding for noisy channels*. Technical Report 335, Research Laboratory of Electronics, MIT, 1957. (Reproduced in *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 5–12, 1991, doi:10.1109/18.61123). 2

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. 6, 12

[GS99] Venkatesan Guruswami and Madhu Sudan. *Improved decoding of Reed-Solomon and algebraic-geometry codes*. IEEE Trans. Inform. Theory, 45(6):1757–1767, 1999. (Preliminary version in *39th FOCS*, 1998). eccc:1998/043. 3, 5, 6, 15

[Kal85] Erich L. Kaltofen. *Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization*. SIAM J. Comput., 14(2):469–489, 1985. 3

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. *Equivalence of polynomial identity testing and polynomial factorization*. Comput. Complexity, 24(2):295–331, 2015. (Preliminary version in *29th CCC*, 2014). eccc:2014/001. 9, 24

[KT22] Itay Kalev and Amnon Ta-Shma. *Unbalanced expanders from multiplicity codes*. In Amit Chakrabarti and Chaitanya Swamy, eds., *Proc. 26th International Conf. on Randomization and Computation (RANDOM)*, volume 245 of *LIPIcs*, pages 12:1–12:14. Schloss Dagstuhl, 2022. eccc:2022/073. 4

[Len85] Arjen K. Lenstra. *Factoring multivariate polynomials over finite fields*. J. Comput. Syst. Sci., 30(2):235–248, 1985. (Preliminary version in *15th STOC*, 1983). 3

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. *Factoring polynomials with rational coefficients*. Math. Ann., 261(4):515–534, 1982. 3

[Sar11] Shubhangi Saraf. *The method of multiplicities*. Ph.D. thesis, Massachusetts Institute of Technology, 2011. 6

[Sho90] Victor Shoup. *On the deterministic complexity of factoring polynomials over finite fields*. Inform. Process. Lett., 33(5):261–267, 1990. 3, 11

[ST21] Amit Sinhababu and Thomas Thierauf. *Factorization of polynomials given by arithmetic branching programs*. Comput. Complexity, 30(2):15, 2021. (Preliminary version in *35th CCC*, 2020). eccc:2020/077. 9, 24, 25

[Sud97] Madhu Sudan. *Decoding of Reed-Solomon codes beyond the error-correction bound*. J. Complexity, 13(1):180–193, 1997. (Preliminary version in *37th FOCS*, 1996). 3, 5, 6

[Sud12] ———. *6.S897: Algebra and Computation*, 2012. (A course on algebraic methods in computation at MIT, Spring 2012). 25

[SW99] Mohammad Amin Shokrollahi and Hal Wasserman. *List decoding of algebraic-geometric codes*. IEEE Trans. Inform. Theory, 45(2):432–437, 1999. 4

[Woz58] John M. Wozencraft. *List decoding.* Quarterly Progress Report, Research Laboratory of Electronics, MIT, 48:90–95, 1958. 2