A Hensel Lifting to Replace Factorization in List-Decoding of Algebraic-Geometric and Reed-Solomon Codes

Daniel Augot, Lancelot Pecquet

A Hensel Lifting to Replace Factorization in List-Decoding of Algebraic-Geometric and Reed-Solomon Codes

Daniel Augot and Lancelot Pecquet

Abstract

This paper presents an algorithmic improvement to Sudan's list--decoding algorithm for Reed-Solomon codes and its generalization to algebraic-geometric codes from Shokrollahi and Wasserman. Instead of completely factoring the interpolation polynomial over the function field of the curve, we compute sufficiently many coefficients of a Hensel development to reconstruct the functions that correspond to codewords. We prove that these Hensel developments can be found efficiently using Newton's method. We also describe the algorithm in the special case of Reed-Solomon codes.

Keywords: List-decoding, algebraic-geometric codes, Reed-Solomon codes, polynomials over algebraic function fields, Hensel lifting, Newton's method.

1 Introduction

In [10, 11], M. Sudan proposed an algorithm of polynomial-time complexity to decode low-rate Reed-Solomon codes [4, p. 294, *sqq.*] beyond the usual correction capability. This problem may have several solutions, thus Sudan's algorithm is a *list-decoding* algorithm. The ideas of this algorithm have been adapted [8] by M. A. Shokrollahi and H. Wasserman to algebraic-geometric (AG) codes. More recently, V. Guruswami and M. Sudan described an enhanced version of the method for all transmission rates [2]. The structure of these algorithms can be sketched as follows:

1. find an interpolation polynomial G under constraints;

2. find some factors of degree 1 of G and retrieve the codewords from them.

In step 2, both algorithms involve factorization or finding the roots of the polynomial G over the function field of an algebraic curve, (which turns to be $\mathbf{F}_q(x)$ in the case of Reed-Solomon codes).

Our contribution is a speedup of this second step, that avoids factorization. We notice that the solutions can be characterized by a finite number of coefficients of their Hensel development and we prove that this development can be computed fast using Newton's method.

For instance, in the case of Reed-Solomon codes, a bivariate polynomial $G(x, T)$ has to be factored over $\mathbf{F}_q$. The method is roughly as follows [13, p. 434]:

1. specialize the variable T in a well-chosen value $y_0 \in \mathbf{F}_q$ in $G(x, T)$;

2. factor the univariate polynomial $G(x, y_0)$;

3. lift the results to get the factorization of $G(x, T) \mod (T - y_0)^l$ for large enough l.

As noted by a referee, both steps 1 and 2 are eliminated in our procedure. The lifting process can be launched immediatly by inspection of the symbols of the received word. This is an improvement since step 1 may require algebraic extensions of the base field [13, p. 433], and since there is no know deterministic algorithm to perform step 2. We use Newton's method to do the lifting, and our algorithm is completely deterministic. The same idea can be adapted in the case of AG codes.

Note that the interpolation step has been investigated by T. Høholdt and R. R. Nielsen [3] and by M. A. Shokrollahi and V. Olshevsky [6].

In Section 2, we recall the list-decoding algorithm of Shokrollahi and Wasserman for algebraic-geometric codes, and the original version of Sudan's algorithm in the special case of Reed-Solomon codes. In Section 3, we recall the main known results concerning Hensel developments of functions in a discrete valuation ring and a method to retrieve such a development using Newton's method when the function is a root of some polynomial. In Section 4, we show how this method can be used to replace the factorization step in Shokrollahi and Wasserman's algorithm. Then in Section 5,

we describe in detail a fast implementation of our method, for which we give some complexity estimates in Section 6. The paper contains three appendices: the first one describes an algorithm to build a basis of functions of the vector space associated to a divisor with increasing valuations at a given place as we use such bases in our implementation. The two following appendices are examples of a step-by-step application of our method to a Reed-Solomon code and to a Hermitian code.

2 Outline of Shokrollahi-Wasserman algorithm

2.1 General idea

We recall the Shokrollahi and Wasserman generalization of Sudan's algorithm.

Let $\mathcal{X}$ be a projective absolutely irreducible curve defined over $K = \mathbf{F}_q$ and let $K(\mathcal{X})$ be its function field. A *place* of $K(\mathcal{X})$ is a maximal ideal of a discrete valuation ring of $K(\mathcal{X})$.

Let $(P_1, \ldots, P_n)$ be n-tuple of pairwise distinct places of degree 1 of $K(\mathcal{X})$, and D be a divisor of $K(\mathcal{X})$ such that none of the P_i is in $\operatorname{Supp} D$, we denote by $\mathcal{L}(D)$ the vector space associated to D and define the evaluation mapping $\mathrm{ev} : \mathcal{L}(D) \to \mathbf{F}_q^n$ by $\mathrm{ev}(f) = (f(P_1), \ldots, f(P_n))$. If $2g - 2 < \deg D < n$, following the terminology of [7], the $[n, k = \ell(D) = \deg D - g + 1, d \geq n - k + 1 - g]_q$-code $C = \mathrm{Im}(f)$ is called a *strongly algebraic-geometric* (SAG) code. The reader may refer to [9, 12] for more complete reference on AG-codes. We henceforth suppose that all these parameters are fixed, and for $1 \leq i \leq n$, we also will denote by l_{P_i} the maximal valuation at place P_i of a function $f \in \mathcal{L}(D)$.

For a given $y \in \mathbf{F}_q^n$, the problem of finding the set $B_\tau(y)$ of all words of C at distance at most τ of y is equivalent to the problem of finding the set $B_\tau^*(y)$ of all functions $f \in \mathcal{L}(D)$ such that $f(P_i) = y_i$ for at least $n - \tau$ values of i. We suppose that y is fixed as well, and rephrase the theorem given in [8]:

Theorem 1 (Shokrollahi and Wasserman, 1999) *Let Δ be a divisor of degree less than $n - \tau$ such that none of the P_i is in $\operatorname{Supp} \Delta$, for all polynomials $G(T) = a_0 + \cdots + a_b T^b \in K(\mathcal{X})[T]$ such that:*

1. *G is nonzero;*

2. *$G(y_i)$ is a function of $K(\mathcal{X})$ vanishing on all P_i, $i \in \{1, \ldots, n\}$;*

3. *a_j is in the space $\mathcal{L}\big(\Delta - jD\big)$ for all $j \in \{0, \ldots, b\}$;*

then for all $f \in B^*_\tau(y)$, f is a root of $G(T)$.

Proof At any place P of $K(\mathcal{X})$, for all $f \in K(\mathcal{X})$, $\mathrm{v}_P(G(f)) \geq \min_{0 \leq j \leq b} \big(\mathrm{v}_P(a_j) + j\,\mathrm{v}_P(f)\big)$. Thus for all j, we have $\mathrm{v}_P(a_j) + j\,\mathrm{v}_P(f) \geq -\mathrm{v}_P(\Delta) + j\,\mathrm{v}_P(D) - j\,\mathrm{v}_P(D) = -\mathrm{v}_P(\Delta)$. Hence $G(f) \in \mathcal{L}(\Delta)$.

Consider the set I of indices for which $f(P_i) = y_i$. For $i \in I$ we have $G(f) = a_0 + \cdots + a_b f^b$. Since $P_i \notin \operatorname{Supp} D \cup \operatorname{Supp} \Delta$, we can evaluate $G(f)$ at P_i. Then, by hypothesis:

$$\big(G(f)\big)(P_i) = \sum_{j=0}^{b} a_j(P_i) f(P_i)^j = \sum_{j=0}^{b} a_j(P_i) y_i^j = \big(G(y_i)\big)(P_i) = 0\ .$$

Consequently $\mathrm{v}_{P_i}\big(G(f)\big) \geq 1$ and $\big(G(f)\big) \geq -\Gamma_I$ where

$$\Gamma_I = \Delta - \sum_{i \in I} P_i\ .$$

The degree of Γ_I is $\deg(\Gamma_I) = \deg(\Delta) - |I|$ and for all $f \in B^*_\tau(y)$, $|I| \geq n - \tau$. Therefore the hypothesis $\deg \Delta < n - \tau$ implies $\deg \Gamma_I < 0$ hence $\mathcal{L}(\Gamma_I) = \{0\}$ and $G(f) = 0$. □

A polynomial G satisfying conditions 1–3 of Theorem 1 exists for all $y \in \mathbf{F}_q^n$ if τ is not too large. From [8], a necessary condition, based on the count of the number of unkowns and linear equations satisfied by G, can be stated as follows:

Theorem 2 *Let C be an $[n, k]$-stronlgy algebraic-geometric code, for any $\tau \leq n + g - \big\lceil \sqrt{2n(k + g - 1)} \big\rceil$, a polynomial satisfying hypothesis of Theorem 1 exists of degree at most $\big\lceil \sqrt{2n/(k + g - 1)} \big\rceil$ for all $y \in \mathbf{F}_q^n$.*

The algorithm is therefore the following:

1. Find a polynomial $G(T)$ satisfying conditions 1–3 of Theorem 1.
2. Compute the roots f of $G(T)$ in $\mathcal{L}(D)$ such that $f(P_i) = y_i$ for at least $n - \tau$ values of i.

2.2 Case of Reed-Solomon codes

The $[n,k]_q$-Reed-Solomon codes can be described as a special case of a strongly algebraic-geometric code. The corresponding curve is the projective line, which is of genus 0, and the function field $K(\mathcal{X})$ is $\mathbf{F}_q(x)$. All places of degree 1 of $\mathbf{F}_q(x)$ are the $P_i = (x - p_i)$ with $p_i \in \mathbf{F}_q$, plus the place at infinity $P_\infty = (1/x)$. The divisor D is $(k-1) \cdot P_\infty$, $\mathcal{L}(D)$ is the set of polynomials of degree less than k over $\mathbf{F}_q$.

Using a divisor $\Delta = \big(n - \tau - 1 - b(k-1)\big) \cdot P_\infty$, Theorem 1 can be reformulated as in Sudan's original articles [10, 11]:

Theorem 3 (Sudan, 1997) *For all bivariate polynomials $G(x,T) = a_0(x) + \cdots + a_b(x)T^b \in \mathbf{F}_q[x,T]$, such that:*

1. *G is nonzero;*
2. *$G(p_i, y_i) = 0$ for all $i \in \{1, \ldots, n\}$;*
3. *$\deg a_j(x) < n - \tau - (k-1)j$ for all $j \in \{0, \ldots, b\}$;*

then for all $f \in B^*_\tau(y)$, $G(x, f(x)) = 0$.

By applying Theorem 2, if $\tau \leq n - \lceil \sqrt{2nk} \rceil$, a polynomial $G(x,T)$ satisfying conditions 1–3 of Theorem 3 exists of degree at most $\lceil \sqrt{2n/k} \rceil$. The algorithm becomes:

1. Find a bivariate polynomial $G(x,T)$ as in Theorem 3.
2. Find factors of the form $(T - f(x))$ where f is a polynomial of $\mathbf{F}_q[x]$ of degree less than k such that $f(p_i) = y_i$ for at least $n - \tau$ values of i.

3 Hensel lifting

In this section we recall some results on Hensel developments and their computation by Newton's method. We will use the following notation until the end of the paper. For a given place P of the function field $K(\mathcal{X})$, we denote by $\mathcal{O}_P$ its discrete valuation ring, by t_P a fixed uniformizer of P, and by v_P the valuation of $\mathcal{O}_P$. Any nonzero element f of $\mathcal{O}_P$ can be written in a

unique way as a converging (*cf.* [1, p. 432]) power series in t_P with coefficients in K, called its HENSEL *development at place* P or P*-adic development*:

$$f = \sum_{j=0}^{\infty} \alpha_j t_P^j \ .$$

The HENSEL development up to order l of f will be denoted by:

$$\mathrm{Hens}_P(f, l) = \sum_{j=0}^{l} \alpha_j t_P^j = f \ \mathrm{rem} \ t_P^{l+1} \ .$$

(To emphasize the algorithmics, we distinguish $f \ \mathrm{rem} \ t_P^{l+1}$, which is a polynomial in t_P, from $f \mod t_P^{l+1}$, which is an element of a quotient ring.) We will also denote by $\mathrm{coef}_P(f, m)$ and $\mathrm{init}_P(f)$, respectively the m-th and the first nonzero coefficient of the development of f.

We recall the following well-known results (see [5, pp. 126, *sqq.*] and [13, pp. 243–263]).

Theorem 4 (Newton's approximation theorem) *Let* $G \in \mathcal{O}_P[T]$ *and* $\varphi_j \in K[t_P]$ *such that* $G(\varphi_j) = 0 \mod t_P^{2^j}$ *and* $G'(\varphi_j) \neq 0 \mod t_P$. *Then* $\varphi_{j+1} = \varphi_j - G(\varphi_j) \cdot G'(\varphi_j)^{-1} \ \mathrm{rem} \ t_P^{2^{j+1}}$ *is defined in* $K[t_P]$ *and*

1. $\varphi_{j+1} = \varphi_j \mod t_P^{2^j}$;
2. $G(\varphi_{j+1}) = 0 \mod t_P^{2^{j+1}}$;
3. $G'(\varphi_{j+1}) \neq 0 \mod t_P$.

Moreover if ψ_{j+1} *is a polynomial in* $K[t_P]$ *such that* $G(\psi_{j+1}) = 0 \mod t_P^{2^{j+1}}$ *and* $\psi_{j+1} = \varphi_{j+1} \mod t_P^{2^j}$, *then* $\psi_{j+1} = \varphi_{j+1} \mod t_P^{2^{j+1}}$ *(unicity of* HENSEL *development).*

Proof As $G'(\varphi_j) \neq 0 \mod t_P$, $G'(\varphi_j) \neq 0 \mod t_P^{2^j}$. In addition $G(\varphi_j) = 0 \mod t_P^{2^j}$ implies $\varphi_{j+1} - \varphi_j = 0 \mod t_P^{2^j}$, which proves assertion 1. TAYLOR's development at order two of $G(T)$ in φ_j is:

$$G(T) = G(\varphi_j) + (T - \varphi_j)G'(\varphi_j) + (T - \varphi_j)^2 R(T - \varphi_j)^2$$

where R is some polynomial in $\mathcal{O}_P[T]$. If we specialize T in φ_{j+1}, we have:

$$\begin{aligned} G(\varphi_{j+1}) &= G(\varphi_j) + (\varphi_{j+1} - \varphi_j)G'(\varphi_j) + (\varphi_{j+1} - \varphi_j)^2 R(\varphi_{j+1} - \varphi_j) \\ &= G(\varphi_j) + (\varphi_{j+1} - \varphi_j)G'(\varphi_j) \mod t_P^{2^{j+1}} \\ &= G(\varphi_j) + \big(- G(\varphi_j) \cdot G'(\varphi_j)^{-1}\big)G'(\varphi_j) \mod t_P^{2^{j+1}} \\ &= 0 \mod t_P^{2^{j+1}} . \end{aligned}$$

This proves assertion 2. Since $\varphi_{j+1} = \varphi_j \mod t_P^{2^j}$, we have $\varphi_{j+1} = \varphi_j \mod t_P$. Consequently, assertion 3. follows since $G'(\varphi_j) \neq 0 \mod t_P$ implies $G'(\varphi_{j+1}) \neq 0 \mod t_P$.

Finally, let ψ_{j+1} be a polynomial in $K[t_P]$ such that $G(\psi_{j+1}) = 0 \mod t_P^{2^{j+1}}$ and $\psi_{j+1} = \varphi_{j+1} \mod t_P^{2^j}$. The second order TAYLOR's development of $G(T)$ in φ_{j+1}, when specialized in ψ_{j+1} is:

$$G(\psi_{j+1}) - G(\varphi_{j+1}) = (\psi_{j+1} - \varphi_{j+1})\big(G'(\varphi_{j+1}) + c \cdot (\psi_{j+1} - \varphi_{j+1})^2\big) ,$$

where $c \in \mathcal{O}_P$. Thus $(\psi_{j+1} - \varphi_{j+1})G'(\varphi_{j+1}) = 0 \mod t_P^{2^{j+1}}$. Since $G'(\varphi_{j+1}) \neq 0 \mod t_P^{2^{j+1}}$, $\psi_{j+1} - \varphi_{j+1} = 0 \mod t_P^{2^{j+1}}$. □

NEWTON*'s method* builds iteratively the HENSEL development of the roots of a given polynomial by repeating the iterations $\varphi_{j+1} \leftarrow \varphi_j - G(\varphi_j) \cdot G'(\varphi_j)^{-1} \text{ rem } t_P^{2^{j+1}}$, as stated in the following corollary:

Corollary 1 *Let $G(T) \in \mathcal{O}_P[T]$ and $\alpha \in K$ such that $(G'(\alpha))(P) \neq 0$. For all $f \in \mathcal{O}_P$ such that $G(f) = 0$ and $f(P) = \alpha$, let $(\varphi_j)_{j\in\mathbf{N}}$ be the sequence defined by $\varphi_0 = \alpha$ and for $j \geq 0$, $\varphi_{j+1} = \varphi_j - G(\varphi_j) \cdot G'(\varphi_j)^{-1} \text{ rem } t_P^{2^{j+1}}$. Then for all $j \in \mathbf{N}$, $\varphi_{j+1} = f \mod t_P^{2^{j+1}}$, and $(\varphi_j)_{j\in\mathbf{N}}$ converges to f in $\mathcal{O}_P$.*

4 Application to Shokrollahi-Wasserman algorithm

Let $G(T)$ ge a polynomial satisfying the conditions of Theorem 1. For $f \in B_\tau^*(y)$, we know that $G(f) = 0$ and in order to use Corollary 1 to build a sequence that converges to f, we need a place P and $\alpha \in K$ such that $f(P) = \alpha$, and $\big(G'(\alpha)\big)(P) \neq 0$. We now prove that if $G(T)$ is chosen of minimal degree, then for any $f \in B_\tau^*(y)$ there is a position i such that $f(P_i) = y_i$ and $\big(G'(y_i)\big)(P) \neq 0$. Thus by choosing $\alpha = y_i$, we will retrieve f using NEWTON's method.

Theorem 5 *If $G(T)$ is a polynomial of minimal degree satisfying the conditions of Theorem 1, then for each root $f \in \mathcal{L}(D)$, there is an index $i \in \{1, \ldots, n\}$ such that $f(P_i) = y_i$ and $\big(G'(y_i)\big)(P_i) \neq 0$.*

Proof Let $a_0 + a_1 T + \cdots + a_b T^b$ be a polynomial of minimal degree, satisfying the conditions of Theorem 1. For any $f \in \mathcal{L}(D)$ such that $G(f) = 0$, let $R(T) = r_0 + \cdots + r_{b-1} T^{b-1} \in K(\mathcal{X})[T]$ be such that $G(T) = (T - f)R(T)$. By identification, we have:

$$G(T) = \underbrace{-f r_0}_{a_0} + \underbrace{(r_0 - f r_1)}_{a_1} T + \cdots + \underbrace{(r_{b-2} - f r_{b-1})}_{a_{b-1}} T^{b-1} + \underbrace{r_{b-1}}_{a_b} T^b \, .$$

Let us prove recursively that $r_j \in \mathcal{L}(D - (j+1)D)$ for $0 \leq j < b$.

It is true for $r_{b-1} = a_b \in \mathcal{L}(D - bD)$. Suppose that for some $j \leq b - 1$, $r_j \in \mathcal{L}(D - (j+1)D)$. Then we have $a_j = r_{j-1} - f r_j$, hence at any place P, $\mathrm{v}_P(r_{j-1}) \geq \min(\mathrm{v}_P(a_j), \mathrm{v}_P(f) + \mathrm{v}_P(r_j))$. On the one hand, we know from condition 3 of Theorem 1 that $\mathrm{v}_P(a_j) \geq -\mathrm{v}_P(\Delta - jD)$; on the other hand, the recursion hypothesis tells that $\mathrm{v}_P(r_j) \geq -\mathrm{v}_P(\Delta - (j+1)D)$. We also know that $f \in \mathcal{L}(D)$, hence $\mathrm{v}_P(f) + \mathrm{v}_P(r_j) \geq -\mathrm{v}_P(\Delta - jD)$. So in all cases, $\mathrm{v}_P(r_{j-1}) \geq -\mathrm{v}_P(\Delta - jD)$.

We therefore have proved that for any $j \leq b - 1$, $r_j \in \mathcal{L}(\Delta - (j+1)D) \subseteq \mathcal{L}(\Delta - jD)$. In particular, $R(T)$ also satisfies condition 3 of Theorem 1.

Consider the set I of indices i such that $f(P_i) = y_i$, then for $i \notin I$, $(R(y_i))(P_i) = 0$. Suppose now that for all $i \in I$, $(R(y_i))(P_i) = 0$, then $R(T)$ would clearly satisfy conditions 1 and 2 of Theorem 1. As we have shown that $R(T)$ satisfies condition 3, it would contradict the fact we have chosen $G(T)$ of minimal degree satisfying these conditions. Consequently, there exists an index i for which $f(P_i) = y_i$ and $\big(R(y_i)\big)(P_i) \neq 0$. The derivative is $G'(T) = (T - f)R'(T) + R(T)$, hence $\big(G'(y_i)\big)(P_i) = 0 + \big(R(y_i)\big)(P_i) \neq 0$. Theorem 1 tells us that any $f \in B^*_\tau(y)$ is a root of $G(T)$. □

As we will see in Section 5.2, any function in $\mathcal{L}(D)$ can be characterized by its HENSEL development up to order l_{P_i}, for any $i \in \{1, \ldots, n\}$. Thus the method to perform step 2 of SHOKROLLAHI and WASSERMAN's algorithm is the following. Iterate through the set S of all positions $i \in \{1, \ldots, n\}$ such that $\big(G'(y_i)\big)(P_i) \neq 0$. For any such i, compute the l_{P_i}-th term of the sequence $(\varphi_j)_{j \in \mathbf{N}}$ as built in Corollary 1 with $\varphi_0 = y_i$ for large enough l_{P_i} (see Section 5.1), then convert $\varphi_{l_{P_i}}$ to a function $f \in \mathcal{L}(D)$, and test if $f \in B_\tau(y)$.

The next section describes all details of this method and the whole algorithm for list-decoding is given in Section 5.3.

5 Implementation

5.1 Newton's method

In NEWTON's method, each iteration $\varphi_{j+1} \leftarrow \varphi_j - G(\varphi_j)/G'(\varphi_j) \text{ rem } t_P^{2^{j+1}}$ involves a division of series, which can be replaced by a multiplication: $\varphi_{j+1} \leftarrow \varphi_j - G(\varphi_j) \cdot \eta_j \text{ rem } t_P^{2^{j+1}}$ where η_j is an auxiliary sequence which approximates $G'(\varphi_j)^{-1}$. More precisely, we define:

Proposition 1 (Newton Inversion) *Let P be a place of degree 1 of $K(\mathcal{X})$, and $h \in K[[t_P]]$ such that $h \neq 0 \mod t_P$. Let $(\eta_j)_{j\in\mathbf{N}}$ be the sequence defined by $\eta_0 = h(P)^{-1}$ and for all $j \in \mathbf{N}$, $\eta_{j+1} = 2\eta_j + h\eta_j^2 \text{ rem } 2^{j+1}$. Then $(\eta_j)_{j\in\mathbf{N}}$ satisfies $\eta_j h = 1 \mod t_P^{2^j}$ for all $j \in \mathbf{N}$.*

Proof For $j = 0$, we have $h\eta_0 = h(P)\eta_0 = 1 \mod t_P^{2^0}$. Suppose the result is true up to j, then $1 - h\eta_{j+1} = 1 - h(2\eta_j - h\eta_j^2) = 1 - 2\eta_j + h^2\eta_j^2 = (1 - h\eta_j)^2 = 0 \mod t_P^{2^{j+1}}$ by the recurrence hypothesis. □

For computational purpose, the polynomial $G(T) = a_0 + \cdots + a_b T^b \in \mathcal{O}_P[T]$ is replaced by the polynomial $\widetilde{G}_{P,l}$ whose coefficients are the HENSEL developments at P of the coefficients of G up to order l, *i.e.* $\widetilde{G}_{P,l}(T) = G(T) \text{ rem } t_P^{l+1} = \mathrm{Hens}_P(a_0, l) + \cdots + \mathrm{Hens}_P(a_b, l) \cdot T^b \in K[t_P][T]$.

Proposition 2 *Let $G(T) \in \mathcal{O}_P[T]$ and $\alpha \in K$ be such that $\big(G'(\alpha)\big)(P) \neq 0$, and let l be a nonnegative integer, then the sequences:*

$$\begin{cases} \eta_0 = \big(\widetilde{G}'_{P,l}(\alpha)\big)(P)^{-1} & \text{and } \forall j \in \mathbf{N},\ \eta_{j+1} = 2\eta_j + \widetilde{G}'_{P,l}(\varphi_j) \cdot \eta_j^2 \text{ rem } t_P^{\min(2^{j+1}, l+1)} \\ \varphi_0 = \alpha & \text{and } \forall j \in \mathbf{N},\ \varphi_{j+1} = \varphi_j - \widetilde{G}_{P,l}(\varphi_j) \cdot \eta_{j+1} \text{ rem } t_P^{\min(2^{j+1}, l+1)} \end{cases}$$

are well-defined and for all $f \in \mathcal{O}_P$ such that $G(f) = 0$ and $f(P) = \alpha$, $\mathrm{Hens}_P(f, l) = \varphi_{\lceil \log_2(l+1) \rceil}$.

Proof First, $\big(\widetilde{G}'_{P,l}(\alpha)\big)(P) = \big(G'(\alpha)\big)(P) \neq 0$, which guarantees that the sequences are well-defined. Let ψ_j be the sequence defined by $\psi_0 = \alpha$ and for all $j \in \mathbf{N}$, $\psi_{j+1} = \psi_j - G(\psi_j) \cdot G'(\psi_j)^{-1} \text{ rem } t^{2^{j+1}}$.

¿From Corollary 1, we know $\psi_{j+1} = f \mod t_P^{2^{j+1}}$ for all $j \in \mathbf{N}$.

Furthermore, as for all $j \in \mathbf{N}$, $G(T) = \widetilde{G}_{P,l}(T) \mod t_P^{\min(2^{j+1}, l+1)}$, we have $\eta_{j+1} = (G'(\varphi_{j+1}))^{-1} \mod t_P^{\min(2^{j+1}, l+1)}$ and $\varphi_{j+1} = \psi_{j+1} \mod t_P^{\min(2^{j+1}, l+1)}$. For $j+1 = \lceil \log_2(l+1) \rceil$, we have $2^{j+1} \geq l+1$, thus $\varphi_{\lceil \log_2(l+1) \rceil} = \psi_{\lceil \log_2(l+1) \rceil} \text{ rem } t_P^{l+1} = f \text{ rem } t_P^{l+1} = \mathrm{Hens}_P(f, l)$. □

We therefore have Algorithm 1 to compute the HENSEL development at P up to order l of a root f of G provided $\big(G'(f(P))\big)(P) \neq 0$.

Algorithm 1

function Newton(G, P, α, l).
Input: A polynomial $G(T) \in \mathcal{O}_P[T]$ and $\alpha \in K$ such that $\big(G(\alpha)\big)(P) = 0$ and $(G'(\alpha))(P) \neq 0$. An integer $l \geq 0$.
Ouput: A polynomial $\varphi \in K[t_P]$ of degree less than or equal to l (φ is equal to $\mathrm{Hens}_P(f, l)$ if there exists $f \in \mathcal{O}_P$ such that $f(P) = \alpha$ and $G(f) = 0$).

1. $\widetilde{G}_{P,l}(T) \leftarrow G(T) \text{ rem } t_P^{l+1}$.
2. $\widetilde{G}'_{P,l}(T) \leftarrow$ the derivative of $\widetilde{G}_{P,l}(T)$.
3. $\eta \leftarrow \big(\widetilde{G}'_{P,l}(\alpha)\big)(P)^{-1}$. *// This is η_0*
4. $\varphi \leftarrow \alpha$. *// Compute φ_0*
5. **for** j **from** 0 **to** $\lceil \log_2(l+1) \rceil - 1$ **do**
   $\eta \leftarrow 2\eta - \widetilde{G}'_{P,l}(\varphi) \cdot \eta^2 \text{ rem } t_P^{\min(2^{j+1}, l+1)}$. *// Compute η_{j+1}*
   $\varphi \leftarrow \varphi - \widetilde{G}_{P,l}(\varphi) \cdot \eta \text{ rem } t_P^{\min(2^{j+1}, l+1)}$. *// Compute φ_{j+1}*
6. **return** φ. *// Return $\varphi_{\lceil \log_2(l+1) \rceil}$*

5.2 Converting Hensel's developments to functions

For our purpose, we focus on the roots of $G(T) \in \mathcal{O}_P[T]$ that are in $\mathcal{L}(D)$ (with $P \notin \mathrm{Supp}\, D$). In general, the output φ of Algorithm 1 is not necessarily the HENSEL development of a function of $\mathcal{L}(D)$. It is possible to decide fast if there exists a function $f \in \mathcal{L}(D)$ whose truncated development is φ, and retrieve f in this case, as described in Algorithm 2. Note that it is possible that f is not a root of $G(T)$. HENSEL's lemma tells us that φ is the truncation of a root of $G(T)$ in the completion of the function field, which may not be in $\mathcal{L}(D)$ *a priori*.

However, when $G(T)$ satisfies condition 3 of Theorem 1, then the output of Algorithm 1 is always the HENSEL development of a function of $\mathcal{L}(D)$, as

shown in the next theorem. Consequently, Algorithm 2 will never return *fail*.

Theorem 6 *Let* $G(T) = a_0 + \cdots + a_b T^b$ *be a polynomial such that* $a_j \in \mathcal{L}(\Delta - jD)$ *for* $0 \leq j \leq b$. *Let* P *be a place of degree* 1 *such that* $P \notin \operatorname{Supp}\Delta \cup \operatorname{Supp} D$, *and* $\alpha \in K$ *be such that* $\big(G(\alpha)\big)(P) = 0$ *and* $\big(G'(\alpha)\big)(P) \neq 0$. *Then with the notation of Proposition 2, for all* $j \in \{0, \ldots, \lceil \log_2(l_P+1) \rceil\}$, $\varphi_{j+1} \in \mathcal{L}(D)$. *Consequently, given the output* **Newton(**G**,** P**,** α**,** l_P**)** *of Algorithm 1 is the* Hensel *development of a function of* $\mathcal{L}(D)$.

Proof We prove this by induction. First, note that $\varphi_0 = \alpha \in \mathcal{L}(D)$ and $\eta_0 = \big(\widetilde{G}'_{P,l_P}(\alpha)\big)(P)^{-1} \in \mathcal{L}(D - \Delta)$. Suppose now that for some $j \in \{0, \ldots, \lceil \log_2(l_P + 1) \rceil\}$, $\varphi_j \in \mathcal{L}(D)$ and $\eta_j \in \mathcal{L}(D - \Delta)$. Then $\widetilde{G}'_{P,l_P}(\varphi_j) \in \mathcal{L}(\Delta - D)$, $\eta_j^2 \in \mathcal{L}(2D - 2\Delta)$, and $\eta_{j+1} = 2\eta_j + \widetilde{G}'_{P,l_P}(\varphi_j) \cdot \eta_j^2 \text{ rem } t_P^{\min(2^{j+1}, l+1)} \in \mathcal{L}(D - \Delta)$. Furthermore, $\widetilde{G}_{P,l_P}(\varphi_j) \in \mathcal{L}(\Delta)$, hence $\widetilde{G}_{P,l_P}(\varphi_j) \cdot \eta_{j+1} \in \mathcal{L}(D)$ so $\varphi_{j+1} = \varphi_j - \widetilde{G}_{P,l_P}(\varphi_j) \cdot \eta_{j+1} \text{ rem } t_P^{\min(2^{j+1}, l+1)} \in \mathcal{L}(D)$. □

We use a special kind of basis of the space $\mathcal{L}(D)$ to retrieve a function in this space from its Hensel development at a given place:

Definition 1 *A basis* $(f_1, \ldots, f_\kappa)$ *of* $\mathcal{L}(D)$ *is said to be in* P-reduced echelon form *if* $\mathrm{v}_P(f_1) < \cdots < \mathrm{v}_P(f_\kappa)$, *and such that* $\mathrm{init}_P(f_i)$ *is* 1 *for all* $i \in \{1, \ldots, \kappa\}$.

The P-valuation sequence *of* D *is the set of valuations* $\mathrm{v}_P(f_1), \ldots, \mathrm{v}_P(f_\kappa)$, *and we denote by by* l_P *the integer* $\max_{f \in \mathcal{L}(D)} \mathrm{v}_P(D) = \mathrm{v}_P(f_\kappa)$.

See Appendix A to find how to construct such a basis from any basis of $\mathcal{L}(D)$. Note that the definition of the valuation sequence does not depend on the choice of the basis in P-reduced echelon form.

We use the result of the following remark in Algorithm 2 to verify if a Hensel development corresponds to a function of $\mathcal{L}(D)$.

Remark 1 *If* $(f_1, \ldots, f_\kappa)$ *is a basis of* $\mathcal{L}(D)$ *in* P*-reduced echelon form, for all nonzero function* $f \in \mathcal{L}(D)$, *if* $f = \lambda_1 f_1 + \cdots + \lambda_\kappa f_\kappa \neq 0$, *then* $\mathrm{v}_P(f) = \mathrm{v}_P(f_i)$ *where* $i = \min\big\{l \in \{1, \ldots, \kappa\} \mid \lambda_l \neq 0\big\}$. *Hence* $\mathrm{v}_P(f) \in \{\mathrm{v}_P(f_1), \ldots, \mathrm{v}_P(f_\kappa)\}$. *In particular,* $l_P = \mathrm{v}_P(f_\kappa)$.

Algorithm 2

function SeriesToFunction(P, φ).
Precomputed: A basis $\mathcal{B}_P = (f_{P,1}, \ldots, f_{P,\kappa})$ of $\mathcal{L}(D)$ in P-reduced echelon form, the integer l_P, and a basis $\widetilde{\mathcal{B}}_P = (\tilde{f}_{P,1}, \ldots, \tilde{f}_{P,\kappa})$ where $\tilde{f}_{P,j} = \mathrm{Hens}_P(f_{P,j}, l_P)$ for $1 \leq j \leq \kappa$. The valuation sequence $V_P = (\mathrm{v}_P(f_{P,1}), \ldots, \mathrm{v}_P(f_{P,\kappa}))$.
Input: $P \notin \operatorname{Supp} D$ is a place of degree 1, φ is a polynomial in $K[t_P]$ of degree less than or equal to l_P.
Ouput: The unique $f \in \mathcal{L}(D)$ such that $\mathrm{Hens}_P(f, l_P) = \varphi$ if such an f exists, *fail* otherwise.

1. Set $f \leftarrow 0$.
2. **repeat**
   if $\varphi \neq 0$ **then**
   $v \leftarrow \mathrm{v}_P(\varphi)$
   if $v \in V$ **then**
   Let i be the index for which $v = V_{P,i}$.
   $\lambda \leftarrow \mathrm{init}_P(\varphi)$.
   $f \leftarrow f + \lambda f_{P,i}$.
   $\varphi \leftarrow \varphi - \lambda \tilde{f}_{P,i}$.
   else $f \leftarrow$ *fail*.
   until $(\varphi = 0)$ **or** $(f =$ *fail*$)$
3. **return** f.

Example 1 *In the case of the rational function field* $\mathbf{F}_q(x)$*, the polynomial* $t = x - a$ *is a uniformizer of the place* $P = (x - a)$ *and a base of* $\mathcal{L}((k-1)P_\infty)$ *in* P*-reduced echelon form is* $(1, (x-a), (x-a)^2, \ldots, (x-a)^{k-1})$*. The* P*-valuation sequence is* $(0, 1, \ldots, k-1)$ *and* $l_P = k - 1$*. Algorithm 2 behaves simply in that case: for a polynomial* $\varphi = a_0 + \cdots + a_{k-1}t^{k-1}$*, it will return the polynomial* $f(x) = \varphi(x - a)$*.*

The following fact is of interest to reduce the cost of the algorithm:

Proposition 3 *Let* S *be the set* $\big\{i \in \{1, \ldots, n\} \mid \big(G'(y_i)\big)(P_i) \neq 0\big\}$*. The set* $\Phi = \{$**SeriesToFunction(**P_i**,** **Newton(**G**,** P_i**,** y_i**,** l_{P_i}**))**$,\ i \in S\}$ *is a set of* m *functions* $f_1, \ldots, f_m \in \mathcal{L}(D)$ *amongst which are all functions of* $B^*_\tau(y)$*. The subsets* $s_j = \{i \in S \mid f_j(P_i) = y_i\}$ *for* $1 \leq j \leq m$ *form a partition of* S*.*

Proof For each position $i \in S$, we know from Theorem 6 that the output φ of Algorithm 1 is the HENSEL development of a function $f =$ **SeriesToFunction(**P_i**,**φ**)** $\in \mathcal{L}(D)$. If two such functions match at some place P_i, Theorem 4 indicates the two HENSEL developments will coincide and therefore the two functions will also coincide. □

This corollary suggests that once a function f has been found, using successively Algorithm 1 then Algorithm 2, it is useless to apply these algorithm at all places P_i with $i \in S$ such that $f(P_i) = y_i$ because they will lead to the same function. Consequently, it is possible to remove the set $I = \{i \in S \mid f(P_i) = y_i\}$ from the set S before to continue to find other functions.

5.3 The whole algorithm

We are now able to compute $B^*_\tau(y)$ using Algorithm 3. We denote by $\mathcal{B}_1, \ldots, \mathcal{B}_n$ some bases in echelon form with respect to the places $P_1, \ldots, P_n$, repectively.

6 Complexity

6.1 General complexity

We denote by $\mathsf{M}(l)$, the cost of the product of two dense power series truncated at order l with coefficients in K. This can be done in $\mathrm{O}(l^2)$ arithmetic

Algorithm 3

ListDecode(y, τ).
Precomputed: For each $i \in \{1, \ldots, n\}$, a basis $\mathcal{B}_{P_i} = (f_{P_i,1}, \ldots, f_{P_i,\kappa})$ of $\mathcal{L}(D)$ in P_i-reduced echelon form, the P_i-valuation sequence V_{P_i}, and a basis $\widetilde{\mathcal{B}}_{P_i} = (\tilde{f}_{P_i,1}, \ldots, \tilde{f}_{P_i,\kappa})$ where $\tilde{f}_{P_i,j} = \mathrm{Hens}_P(f_{P_i,j}, l_{P_i})$ for $1 \leq j \leq \kappa$.
Input: $y \in \mathbf{F}_q^n$, and τ such that Theorem 1 applies.
Ouput: The set $B_\tau(y) = \{c \in C \mid \mathrm{d}(c, y) \leq \tau\}$.

1. $G(T) \leftarrow$ a polynomial of minimal degree satisfying the conditions of Theorem 1.
2. Set $B \leftarrow \varnothing$.
3. Set $S \leftarrow \big\{i \in \{1, \ldots, n\} \;\big|\; \big(G(y_i)\big)'(P_i) \neq 0\big\}$.
4. **for** i **in** S **do**
   a) $\varphi \leftarrow$ **Newton**(G, P_i, y_i, l_{P_i}). *// In* $\mathcal{L}(D)$
   b) $f \leftarrow$ **SeriesToFunction**(P_i, φ). *// Never fails*
   c) $c \leftarrow \mathrm{ev}(f)$ *// Well defined since* $f \in \mathcal{L}(D)$.
   d) $I \leftarrow \big\{j \in \{1, \ldots, n\} \;\big|\; f(P_j) = y_j\big\}$.
   e) $S \leftarrow S \setminus I$. *// The indices in* I *would lead to the same function*
   f) **if** $|I| \geq n - \tau$ **then** *// Corresponds to a codeword in* $B_\tau(y)$
      Include c into the set B.
5. **return** B.

operations in K with standard polynomial multiplication, $O(l^{1.59})$ operations with KARATSUBA multiplication and $O(l \log l)$ operations with a Fast FOURIER Transform (which may require a field extension in order to get a primitive l-th root of unity).

Almost all operations of Algorithm 3 involve the computation of HENSEL developments, valuations and evaluations of functions of $K(\mathcal{X})$. The cost of these operations depend a lot on the implementation of the function field. However, we can give a cost of the main loop of Algorithm 1:

Proposition 4 *The main loop of Algorithm 1 can be performed at place P in deterministic $O(b\mathsf{M}(l_P))$ arithmetic operations in K.*

Proof First, note that all operations of the kind u rem t_P^m do not cost anything since they just mean not to compute terms of valuation greater than or equal to m. Using HORNER's rule [13, p. 93], we can compute $\widetilde{G}_{P,l}(\varphi)$ and $\widetilde{G}_{P,l}(\varphi)$ in b multiplications and additions of truncated power series at order 2^{j+1}, this can be performed in $O(b\mathsf{M}(2^{j+1}))$ operations. The other operations to compute η and φ are also multiplications and additions of truncated power series at order 2^{j+1} hence the cost of one iteration is $O(b\mathsf{M}(2^{j+1}))$ operations. By convexity of the cost function M, we have $\mathsf{M}(2^{j+1}) \geq 2\mathsf{M}(2^j)$, let us denote by N the integer $\lceil \log_2 l_P + 1 \rceil$, the global cost is: $O(b(\mathsf{M}(1) + \cdots + \mathsf{M}(2^N))) = O(b(\mathsf{M}(2^N)/2^N + \mathsf{M}(2^N)/2^{N-1} + \cdots + \mathsf{M}(2^N))) = O(b\mathsf{M}(2^N)) = O(b\mathsf{M}(l_P))$ operations. □

6.2 Case of Reed-Solomon codes

We fix a transmission rate $R = k/n$, and study the asymptotic behaviour of the algorithm when n goes to infinity.

Proposition 5 *For an $[n, Rn]_q$-REED-SOLOMON code, step 2 of SUDAN's algorithm can be performed deterministically in $O(n\mathsf{M}(n) \log n)$ arithmetic operations in $\mathbf{F}_q$ and even in $O(n^2 \log n)$ operations using a Fast FOURIER Transform.*

Proof For each coefficient $a_j(x)$ of $G(T)$, we can compute all $a_j(x - p_i)$ for $1 \leq i \leq n$ in $O(\mathsf{M}(n) \log n)$ operations using fast multipoint evaluation/interpolation (*cf.* [13, p. 279–285]). This can also be done in $O(n \log n)$ operations using a Fast FOURIER Transform (FFT).

The global cost of the computation of $\widetilde{G}_{P_i,k-1}(T)$ for $1 \leq i \leq n$ is therefore $\mathrm{O}(b\mathsf{M}(n)\log n)$ operations without a FFT and $\mathrm{O}(bn\log n)$ operations using a FFT.

For a given i, in the NEWTON step (Algorithm 1), the computation of derivatives can be done in $\mathrm{O}(bn)$ operations and the computation of η requires $\mathrm{O}(bn)$ operations, then from Proposition 4, we perform the loop in $\mathrm{O}(b\mathsf{M}(n))$ operations. The back-translation (Algorithm 2) can also be done by fast multipoint evaluation/interpolation in $\mathrm{O}(\mathsf{M}(n)\log n)$ operations or in $\mathrm{O}(n\log n)$ operations using a FFT. Steps c) and d) are free as we have evaluated f at all p_i, $1 \leq i \leq n$. Step e) can be done in $\mathrm{O}(n)$ operations.

The degree in T of the polynomial $G(x,T)$ is bounded above by $\sqrt{2/R}+1$, and we can consider that b is constant. The **for** loop is run n times in the worst case, therefore the global cost is $\mathrm{O}(n\mathsf{M}(n)\log n)$ operations without a FFT, and $\mathrm{O}(n^2\log n)$ operations using a FFT. □

Invoking the result of [6], we know that step 1 of Algorithm 3 in which one builds the polynomial $G(T)$ can be performed in $\mathrm{O}(n^2)$ operations. In the case of REED-SOLOMON codes, the FFT is feasible, thus we can conclude:

Corollary 2 *The list-decoding of an* $[n, Rn]_q$-REED-SOLOMON *code can be performed deterministically using* SUDAN*'s algorithm in* $\mathrm{O}(n^2\log n)$ *arithmetic operations in* $\mathbf{F}_q$.

7 Conclusion

We have presented an algorithm to find specific roots of polynomials with coefficients in the function field of an algebraic curve. This algorithm is applied to the list decoding algorithm of AG codes designed by SHOKROLLAHI and WASSERMAN, where it is needed to find roots which belong to the space $\mathcal{L}(D)$ defining the AG code. The algorithm computes HENSEL developments of possible solutions, using NEWTON's method. Assuming that the polynomial to factor has *minimal degree* with respect to the conditions imposed by SHOKROLLAHI and WASSERMAN's algorithm, initial values can be found to start the iterations. This removes the step of univariate factorizations in root-finding algorithms. As a consequence our algorithm is completly deterministic and fast. We also note that it very easy to implement with a very simple control structure (one loop), and that it avoids the use of algebraic extensions of the base field.

Our algorithm can be applied to any AG codes, whether the curve is singular or not. Since many operations are dependent on manipulations of functions on the curve, we cannot give a general complexity measure. But at least we have proven an quadratic complexity up to logarithmic factors in the case of Reed-Solomon codes.

For our algorithm to work, we need that the polynomial $G(T)$ is of minimal degree. The algorithm can not be applied to the more powerful generalisation of Guruswami and Sudan [2], where multiplicities are imposed on the polynomial $G(T)$. An adaptation of this algorithm will be presented in a future paper.

8 Acknowledgement

The authors would like to thank the two anonymous referees who have sent a very complete report on our work, including many helpful remarks.

A Computation of reduced echelon form bases.

The construction of a basis of $\mathcal{L}(D)$ in P-reduced echelon form, as described in Algorithm 4, corresponds to the construction of a matrix in reduced echelon form from the matrix of the P-adic expansions of the f_i, up to order l_P.

B A $[17, 5, 13]_{17}$ extended Reed-Solomon code

We consider the field $K = \mathbf{F}_{17}$. A basis of the vector space $\mathcal{L}(4P_\infty)$, of polynomials of degree less than 5 is $(f_1 = 1, f_2 = x, f_3 = x^2, f_4 = x^3, f_5 = x^4)$.

We build the extended Reed-Solomon code of dimension 5 by evaluating polynomials of degree less than 5 on points $p_1 = 0, p_2 = 1, \ldots, p_n = 16$. Its actual minimum distance is $n - k + 1 = 13$. Consequently, its usual correction capability is $t = \lfloor (d-1)/2 \rfloor = 6$.

A generator matrix of C in reduced echelon form is:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 5 & 15 & 1 & 2 & 7 & 6 & 7 & 2 & 1 & 15 & 5 \\
0 & 1 & 0 & 0 & 0 & 12 & 10 & 15 & 10 & 8 & 1 & 11 & 5 & 14 & 14 & 11 & 7 \\
0 & 0 & 1 & 0 & 0 & 10 & 11 & 7 & 8 & 13 & 10 & 10 & 13 & 8 & 7 & 11 & 10 \\
0 & 0 & 0 & 1 & 0 & 7 & 11 & 14 & 14 & 5 & 11 & 1 & 8 & 10 & 15 & 10 & 12 \\
0 & 0 & 0 & 0 & 1 & 5 & 15 & 1 & 2 & 7 & 6 & 7 & 2 & 1 & 15 & 5 & 1
\end{pmatrix}$$

Algorithm 4

procedure ReducedEchelonForm($\mathcal{B}$, P).
Input: A basis $\mathcal{B} = (f_1, \ldots, f_\kappa)$ of $\mathcal{L}(D)$ and a place P.
Specification: Transforms $\mathcal{B}$ in P-echelon form.

1. $to\_do \leftarrow [1, \ldots, \kappa]$
2. **while** $|to\_do| > 0$ **do**
   a) Let v be the minimal valuation of all functions f_i $i \in to\_do$.
   b) Let i be the minimum index for which $\mathrm{v}_P(f_i) = v$.
   c) $f_i \leftarrow f_i / \operatorname{init}_P(f_i)$.
   d) Remove i from to_do.
   e) $s \leftarrow \{ j \in to\_do \mid \mathrm{v}_P(f_j) = \mathrm{v}_P(f_i) \}$.
   f) **for** $j \in s$ **do** $f_j \leftarrow f_j - \operatorname{init}_P(f_j) \cdot f_i$.
3. Sort $\mathcal{B}$ with respect to increasing valuations.

According to Theorem 2, we can correct up to $\tau = 5$ errors. In reality, the algorithm gives decoding up to $\tau = 7$ errors. Consider now a vector $y = (10, 6, 0, 16, 11, 0, 4, 8, 10, 9, 4, 0, 14, 9, 11, 12, 15)$. A polynomial satisfying the conditions of Theorem 3 of minimal degree is $G(x, T) = xT^2 + (11x^4 + 10x^3 + 7x^2 + 12x)T + 15x^9 + 12x^8 + 3x^7 + 10x^6 + 8x^5 + 7x^4 + 7x^3 + x^2 + x$, and $(\partial G/\partial T)(x, T)$ does not vanish on (p_i, y_i) for $i \in S = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$.

Consider position $i = 2$. We take the basis of the vector space of polynomials of degree less than $k = 5$ consisting in powers of $t_{P_2} = (x - 1)$, namely: $(f_{P_2,1} = 1, f_{P_2,2} = x+16, f_{P_2,3} = x^2+15x+1, f_{P_2,4} = x^3+14x^2+3x+16, f_{P_2,5} = x^4 + 13x^3 + 6x^2 + 13x + 1)$. We now compute $\widetilde{G}_{P_2,4} = G(x-1) = (1+t_{P_2})T^2 + (6+15t_{P_2}+t_{P_2}^2+3t_{P_2}^3+11t_{P_2}^4)T + 13 + 13t_{P_2} + 9t_{P_2}^2 + 6t_{P_2}^3 + 6t_{P_2}^4$.

We start NEWTON's method with $\alpha = y_2 = 6$. The derivative of $\widetilde{G}_{P_2,4}$ is $\widetilde{G}'_{P_2,4} = (2 + 2t_{P_2})T + 6 + 15t_{P_2} + t_{P_2}^2 + 3t_{P_2}^3 + 11t_{P_2}^4$. We initialize:

- $\eta \leftarrow \widetilde{G}'_{P_2,4}(\alpha)^{-1} = 1$

- $\varphi \leftarrow \alpha = 6$

We have to loop $\lceil \log_2(l_{P_2} + 1) \rceil = 3$ times and:

For $j = 0$:

- $\eta = 1 + 7t_{P_2}$
- $\varphi = 6 + 14t_{P_2}$

For $j = 1$:

- $\eta = 1 + 13t_{P_2} + 2t_{P_2}^2 + 7t_{P_2}^3$
- $\varphi = 6 + 14t_{P_2} + 6t_{P_2}^2 + 14t_{P_2}^3$

For $j = 2$:

- $\eta = 1 + 13t_{P_2} + 9t_{P_2}^2 + 4t_{P_2}^4$
- $\varphi = 6 + 14t_{P_2} + 6t_{P_2}^2 + 14t_{P_2}^3 + 11t_{P_2}^4$

The truncated power series φ is the HENSEL development of a function $f = 6 \cdot f_{P_2,1} + 14 \cdot f_{P_2,2} + 6 \cdot f_{P_2,3} + 14 \cdot f_{P_2,4} + 11 \cdot f_{P_2,5} = 11x^4 + 4x^3 + 13x^2 + 12$. We have $\mathrm{ev}(f) = (12, 6, 0, 6, 11, 11, 11, 8, 8, 9, 1, 0, 14, 9, 11, 4, 15)$, which is at distance $\mathrm{d}(c, y) = 7 \leq \tau$ from y, hence we include it in the list B. Moreover, we know that we can remove the set of indices $I = \{j \in S \mid f(P_j) = y_j\} = \{2, 3, 5, 8, 10, 12, 13, 14, 15\}$ from our investigation.

Consider position $i = 4$. We take the basis of the vector space of polynomials of degree less than $k = 5$ consisting in powers of $t_{P_4} = (x - 3)$, namely: $(f_{P_4,1} = 1, f_{P_4,2} = x + 14, f_{P_4,3} = x^2 + 11x + 9, f_{P_4,4} = x^3 + 8x^2 + 10x + 7, f_{P_4,5} = x^4 + 5x^3 + 3x^2 + 11x + 13)$. We now compute $\widetilde{G}_{P_4,4} = G(x - 3) = (3 + t_{P_4})T^2 + (2 + 16t_{P_4} + 11t_{P_4}^2 + 6t_{P_4}^3 + 11t_{P_4}^4)T + 16 + 16t_{P_4} + 3t_{P_4}^2 + t_{P_4}^3 + 15t_{P_4}^4$.

We start NEWTON's method with $\alpha = y_4 = 16$. The derivative of $\widetilde{G}_{P_4,4}$ is $\widetilde{G}'_{P_4,4} = (6 + 2t_{P_4})T + 2 + 16t_{P_4} + 11t_{P_4}^2 + 6t_{P_4}^3 + 11t_{P_4}^4$. We initialize:

- $\eta \leftarrow \widetilde{G}'_{P_4,4}(\alpha)^{-1} = 4$
- $\varphi \leftarrow \alpha = 16$

We have to loop $\lceil \log_2(l_{P_4}+1) \rceil = 3$ times and:

For $j=0$:

- $\eta = 4 + 14t_{P_4}$
- $\varphi = 16 + 13t_{P_4}$

For $j=1$:

- $\eta = 4 + 7t_{P_4} + 3t_{P_4}^2 + 15t_{P_4}^3$
- $\varphi = 16 + 13t_{P_4} + 13t_{P_4}^2 + 6t_{P_4}^3$

For $j=2$:

- $\eta = 4 + 7t_{P_4} + 4t_{P_4}^2 + 6t_{P_4}^4$
- $\varphi = 16 + 13t_{P_4} + 13t_{P_4}^2 + 6t_{P_4}^3 + 6t_{P_4}^4$

The truncated power series φ is the HENSEL development of a function $f = 16 \cdot f_{P_4,1} + 13 \cdot f_{P_4,2} + 13 \cdot f_{P_4,3} + 6 \cdot f_{P_4,4} + 6 \cdot f_{P_4,5} = 6x^4 + 2x^3 + 11x^2 + 10x + 10$. We have $\mathrm{ev}(f) = (10, 5, 16, 16, 3, 0, 4, 3, 10, 12, 4, 6, 12, 7, 1, 12, 15)$, which is at distance $\mathrm{d}(c, y) = 9 > \tau$ from y. So we discard this candidate.

Moreover, we know that we can remove the set of indices $I = \{j \in S \mid f(P_j) = y_j\} = \{4, 6, 7, 9, 11, 16\}$ from our investigation.

Finally, the list contains 1 codeword, namely:

- $c_1 = (12, 6, 0, 6, 11, 11, 11, 8, 8, 9, 1, 0, 14, 9, 11, 4, 15)$

C A $[64, 3, 29]_{16}$ Hermitian code

We consider the field $K = \mathbf{F}_{16} = \mathbf{F}_2[\omega]$ where the primitive element ω has minimal polynomial $\omega^4 + \omega + 1$. We consider the absolutely irreducible affine curve $\mathcal{X}$ with defining polynomial $X^5 + Y^4 + Y$ over K. Its genus is $g = 6$. The curve has 64 points of degree 1, namely: $p_1 = (0, 0)$, $p_2 = (0, 1)$, $p_3 = (0, \omega^5)$, $p_4 = (0, \omega^{10})$, $p_5 = (1, \omega)$, $p_6 = (1, \omega^2)$, $p_7 = (1, \omega^4)$, $p_8 = (1, \omega^8)$, $p_9 = (\omega, \omega^6)$, $p_{10} = (\omega, \omega^7)$, $p_{11} = (\omega, \omega^9)$, $p_{12} = (\omega, \omega^{13})$, $p_{13} = (\omega^2, \omega^3)$, $p_{14} = (\omega^2, \omega^{11})$, $p_{15} = (\omega^2, \omega^{12})$, $p_{16} = (\omega^2, \omega^{14})$, $p_{17} = (\omega^3, \omega)$,

$p_{18} = (\omega^3, \omega^2)$, $p_{19} = (\omega^3, \omega^4)$, $p_{20} = (\omega^3, \omega^8)$, $p_{21} = (\omega^4, \omega^6)$, $p_{22} = (\omega^4, \omega^7)$, $p_{23} = (\omega^4, \omega^9)$, $p_{24} = (\omega^4, \omega^{13})$, $p_{25} = (\omega^5, \omega^3)$, $p_{26} = (\omega^5, \omega^{11})$, $p_{27} = (\omega^5, \omega^{12})$, $p_{28} = (\omega^5, \omega^{14})$, $p_{29} = (\omega^6, \omega)$, $p_{30} = (\omega^6, \omega^2)$, $p_{31} = (\omega^6, \omega^4)$, $p_{32} = (\omega^6, \omega^8)$, $p_{33} = (\omega^7, \omega^6)$, $p_{34} = (\omega^7, \omega^7)$, $p_{35} = (\omega^7, \omega^9)$, $p_{36} = (\omega^7, \omega^{13})$, $p_{37} = (\omega^8, \omega^3)$, $p_{38} = (\omega^8, \omega^{11})$, $p_{39} = (\omega^8, \omega^{12})$, $p_{40} = (\omega^8, \omega^{14})$, $p_{41} = (\omega^9, \omega)$, $p_{42} = (\omega^9, \omega^2)$, $p_{43} = (\omega^9, \omega^4)$, $p_{44} = (\omega^9, \omega^8)$, $p_{45} = (\omega^{10}, \omega^6)$, $p_{46} = (\omega^{10}, \omega^7)$, $p_{47} = (\omega^{10}, \omega^9)$, $p_{48} = (\omega^{10}, \omega^{13})$, $p_{49} = (\omega^{11}, \omega^3)$, $p_{50} = (\omega^{11}, \omega^{11})$, $p_{51} = (\omega^{11}, \omega^{12})$, $p_{52} = (\omega^{11}, \omega^{14})$, $p_{53} = (\omega^{12}, \omega)$, $p_{54} = (\omega^{12}, \omega^2)$, $p_{55} = (\omega^{12}, \omega^4)$, $p_{56} = (\omega^{12}, \omega^8)$, $p_{57} = (\omega^{13}, \omega^6)$, $p_{58} = (\omega^{13}, \omega^7)$, $p_{59} = (\omega^{13}, \omega^9)$, $p_{60} = (\omega^{13}, \omega^{13})$, $p_{61} = (\omega^{14}, \omega^3)$, $p_{62} = (\omega^{14}, \omega^{11})$, $p_{63} = (\omega^{14}, \omega^{12})$, $p_{64} = (\omega^{14}, \omega^{14})$, Its projective closure also contains the point $p_\infty = (0 : 1 : 0)$.

All points are smooth and we denote by P_i the place corresponding to p_i for $1 \leq i \leq 64$, and by P_∞ the place over the point p_∞.

We define the divisor $D = 7 \cdot P_\infty$. A basis of the Riemann-Roch space $\mathcal{L}(D)$ is $(f_1 = 1, f_2 = X, f_3 = Y)$.

We build the Hermitian AG-code with support $(P_1, \ldots, P_n)$ and divisor D. It is a $[64, 3]$-code. Its Goppa designed distance is 57 and its actual minimum distance is 59. Consequently, its usual correction capability is $t = \lfloor (d-1)/2 \rfloor = 29$.

According to Theorem 2, we can correct up to $\tau = 28$ errors. In reality, the algorithm gives decoding up to $\tau = 31$ errors. Consider now a vector $y = (\omega^7, 0, \omega^2, \omega^{12}, 0, \omega^2, 0, \omega^{12}, \omega^6, \omega^3, \omega^{11}, \omega^{10}, \omega^7, \omega^7, \omega, 0, \omega^4, 0, \omega^7, \omega^{14}, \omega^{14}, \omega^{13}, \omega^3, \omega, 1, \omega^{10}, \omega^{11}, \omega^{13}, \omega^{10}, \omega^3, 1, \omega, \omega^{10}, \omega^4, \omega^3, \omega^6, \omega^{11}, \omega^{13}, \omega^2, \omega^8, \omega^{12}, \omega^8, \omega^5, \omega^8, \omega^{11}, \omega^9, \omega^4, \omega^{14}, 1, 0, \omega^7, \omega^2, \omega, 1, \omega^7, \omega^3, \omega^{14}, \omega^{11}, \omega^8, 1, \omega^{10}, \omega^4, \omega^4, \omega^{14})$.

A polynomial satisfying the conditions of Theorem 1 of minimal degree is $G(T) = \omega^{13}X^4T^2 + (\omega^2X^5 + \omega^{14}X^4Y + \omega^4X^4)T + \omega^{11}X^6 + X^5Y + \omega^8X^5 + \omega^4X^4Y^2 + \omega X^4Y + X^4$, and its derivative does not vanish on P_i for $i \in S = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64\}$.

Consider position $i = 5$. A local parameter at place P_5 is $t_{P_5} = (X - 1)$. A basis of $\mathcal{L}(D)$ in P_5-reduced echelon form is $(f_{P_5,1} = 1, f_{P_5,2} = X + 1, f_{P_5,3} = X + Y + \omega^4)$. The P_5-valuation sequence is $V_{P_5} = (0, 1, 5)$, and we have $l_{P_5} = 5$.

We now compute $\widetilde{G}_{P_5,5} = (\omega^{13} + \omega^{13}t_{P_5}^4)T^2 + (\omega^5 + \omega^{13}t_{P_5} + \omega^5 t_{P_5}^4 + \omega^2 t_{P_5}^5)T + \omega^2 t_{P_5} + \omega^6 t_{P_5}^2 + \omega^{10} t_{P_5}^5$.

We start NEWTON's method with $\alpha = y_5 = 0$. The derivative of $\widetilde{G}_{P_5,5}$ is $\widetilde{G}'_{P_5,5} = \omega^5 + \omega^{13} t_{P_5} + \omega^5 t_{P_5}^4 + \omega^2 t_{P_5}^5$. We initialize:

- $\eta \leftarrow \widetilde{G}'_{P_5,5}(\alpha)^{-1} = \omega^{10}$
- $\varphi \leftarrow \alpha = 0$

We have to loop $\lceil \log_2(l_{P_5} + 1) \rceil = 3$ times and:
For $j = 0$:

- $\eta = \omega^{10} + \omega^3 t_{P_5}$
- $\varphi = \omega^{12} t_{P_5}$

For $j = 1$:

- $\eta = \omega^{10} + \omega^3 t_{P_5} + \omega^{11} t_{P_5}^2 + \omega^4 t_{P_5}^3$
- $\varphi = \omega^{12} t_{P_5}$

For $j = 2$:

- $\eta = \omega^{10} + \omega^3 t_{P_5} + \omega^{11} t_{P_5}^2 + \omega^4 t_{P_5}^3 + \omega^3 t_{P_5}^4 + \omega^{13} t_{P_5}^5$
- $\varphi = \omega^{12} t_{P_5} + \omega^{14} t_{P_5}^5$

The truncated power series φ is the HENSEL development of a function $f = \omega^{12} \cdot f_{P_5,2} + \omega^{14} \cdot f_{P_5,3} = \omega^5 X + \omega^{14} Y + \omega^{10}$. We have $\mathrm{ev}(f) = (\omega^{10}, \omega^{11}, \omega^2, \omega^{13}, 0, \omega^4, \omega^{14}, \omega^9, \omega^{13}, \omega^{10}, \omega^{11}, \omega^2, \omega^3, \omega^7, \omega, 1, \omega^4, 0, \omega^9, \omega^{14}, \omega^7, 1, \omega^3, \omega, \omega^2, \omega^{10}, \omega^{11}, \omega^{13}, \omega^3, \omega^7, 1, \omega, \omega^{11}, \omega^2, \omega^{13}, \omega^{10}, \omega^{11}, \omega^{13}, \omega^2, \omega^{10}, \omega^{12}, \omega^6, \omega^5, \omega^8, 0, \omega^9, \omega^4, \omega^{14}, 1, \omega, \omega^7, \omega^3, \omega, 1, \omega^7, \omega^3, \omega^{14}, \omega^4, \omega^9, 0, 0, \omega^4, \omega^9, \omega^{14})$, which is at distance $\mathrm{d}(c, y) = 31 \leq \tau$ from y, hence we include it in the list B. Moreover, we know that we can remove the set of indices $I = \{j \in S \mid f(P_j) = y_j\} = \{5, 11, 14, 15, 17, 18, 20, 23, 26, 27, 28, 31, 32, 38, 39, 41, 43, 44, 47, 48, 49, 53, 54, 55, 56, 57, 62, 64\}$ from our investigation.

Consider position $i = 6$. A local parameter at place P_6 is $t_{P_6} = (X-1)$. A basis of $\mathcal{L}(D)$ in P_6-reduced echelon form is $(f_{P_6,1} = 1, f_{P_6,2} = X+1, f_{P_6,3} = X+Y+\omega^8)$. The P_6-valuation sequence is $V_{P_6} = (0, 1, 5)$, and we have $l_{P_6} = 5$.

We now compute $\widetilde{G}_{P_6,5} = (\omega^{13}+\omega^{13}t_{P_6}^4)T^2+(\omega^8+\omega^{13}t_{P_6}+\omega^8t_{P_6}^4+\omega^2t_{P_6}^5)T+\omega^4+\omega t_{P_6}+\omega^6t_{P_6}^2+\omega^4t_{P_6}^4+t_{P_6}^5$.

We start Newton's method with $\alpha = y_6 = \omega^2$. The derivative of $\widetilde{G}_{P_6,5}$ is $\widetilde{G}'_{P_6,5} = \omega^8+\omega^{13}t_{P_6}+\omega^8t_{P_6}^4+\omega^2t_{P_6}^5$. We initialize:

- $\eta \leftarrow \widetilde{G}'_{P_6,5}(\alpha)^{-1} = \omega^7$
- $\varphi \leftarrow \alpha = \omega^2$

We have to loop $\lceil \log_2(l_{P_6}+1) \rceil = 3$ times and:

For $j = 0$:

- $\eta = \omega^7+\omega^{12}t_{P_6}$
- $\varphi = \omega^2+\omega^{11}t_{P_6}$

For $j = 1$:

- $\eta = \omega^7+\omega^{12}t_{P_6}+\omega^2t_{P_6}^2+\omega^7t_{P_6}^3$
- $\varphi = \omega^2+\omega^{11}t_{P_6}$

For $j = 2$:

- $\eta = \omega^7+\omega^{12}t_{P_6}+\omega^2t_{P_6}^2+\omega^7t_{P_6}^3+\omega^2t_{P_6}^4+\omega^5t_{P_6}^5$
- $\varphi = \omega^2+\omega^{11}t_{P_6}+\omega^7t_{P_6}^5$

The truncated power series φ is the Hensel development of a function $f = \omega^2 \cdot f_{P_6,1}+\omega^{11} \cdot f_{P_6,2}+\omega^7 \cdot f_{P_6,3} = \omega^8X+\omega^7Y+\omega^7$. We have $\mathrm{ev}(f) = (\omega^7,0,\omega^2,\omega^{12},\omega^7,\omega^2,0,\omega^{12},\omega^6,\omega^3,\omega^4,\omega^{10},\omega^7,\omega^2,\omega^{12},0,0,\omega^{12},\omega^7,\omega^2,\omega^{14},\omega^{13},\omega^5,\omega,1,\omega^{11},\omega^8,\omega^9,\omega^{10},\omega^3,\omega^6,\omega^4,\omega^{10},\omega^4,\omega^3,\omega^6,\omega^{11},1,\omega^9,\omega^8,\omega^9,\omega^8,1,\omega^{11},\omega^{11},\omega^9,1,\omega^8,\omega^{12},0,\omega^7,\omega^2,\omega^3,\omega^{10},\omega^4,\omega^6,\omega^9,\omega^{11},\omega^8,1,\omega^{10},\omega^3,\omega^4,\omega^6)$, which is at distance $\mathrm{d}(c,y) = 28 \leq \tau$ from y, hence we include it in the list B. Moreover, we know that we can remove the set of indices $I = \{j \in S \mid f(P_j) = y_j\} = \{6,7,8,9,10,12,13,16,19,21,22,25,29,30,33,34,35,36,40,42,45,50,52,58,59,60,61,63\}$ from our investigation.

Finally, the list contains 2 codewords, namely:

- $c_1 = (\omega^7, 0, \omega^2, \omega^{12}, \omega^7, \omega^2, 0, \omega^{12}, \omega^6, \omega^3, \omega^4, \omega^{10}, \omega^7, \omega^2, \omega^{12}, 0, 0, \omega^{12}, \omega^7, \omega^2, \omega^{14}, \omega^{13}, \omega^5, \omega, 1, \omega^{11}, \omega^8, \omega^9, \omega^{10}, \omega^3, \omega^6, \omega^4, \omega^{10}, \omega^4, \omega^3, \omega^6, \omega^{11}, 1, \omega^9, \omega^8, \omega^9, \omega^8, 1, \omega^{11}, \omega^{11}, \omega^9, 1, \omega^8, \omega^{12}, 0, \omega^7, \omega^2, \omega^3, \omega^{10}, \omega^4, \omega^6, \omega^9, \omega^{11}, \omega^8, 1, \omega^{10}, \omega^3, \omega^4, \omega^6)$

- $c_2 = (\omega^{10}, \omega^{11}, \omega^2, \omega^{13}, 0, \omega^4, \omega^{14}, \omega^9, \omega^{13}, \omega^{10}, \omega^{11}, \omega^2, \omega^3, \omega^7, \omega, 1, \omega^4, 0, \omega^9, \omega^{14}, \omega^7, 1, \omega^3, \omega, \omega^2, \omega^{10}, \omega^{11}, \omega^{13}, \omega^3, \omega^7, 1, \omega, \omega^{11}, \omega^2, \omega^{13}, \omega^{10}, \omega^{11}, \omega^{13}, \omega^2, \omega^{10}, \omega^{12}, \omega^6, \omega^5, \omega^8, 0, \omega^9, \omega^4, \omega^{14}, 1, \omega, \omega^7, \omega^3, \omega, 1, \omega^7, \omega^3, \omega^{14}, \omega^4, \omega^9, 0, 0, \omega^4, \omega^9, \omega^{14})$

References

[1] N. Bourbaki. *Commutative Algebra, Chapters 1–7*. Springer-Verlag, 1989.

[2] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon codes and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 6(45):1757–1767, 1999.

[3] T. Høholdt and R. R. Nielsen. Decoding Hermitian codes with Sudan's algorithm. In *Proceedings of AAECC-13*, LNCS 1719, pages 260–270. Springer-Verlag, 1999.

[4] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting codes*. North-Holland Mathematical Library. North-Holland, 1988.

[5] J. Neukirch. *Algebraic Number Theory*. Springer-Verlag, 1999.

[6] V. Olshevsky and A. Shokrollahi. A displacement structure approach to efficient decoding of Reed-Solomon and algebraic-geometric codes. In *Proceedings of STOC 99*, 1999. to appear.

[7] R. Pellikaan, B. Z. Shen, and G. J. M. van Wee. Which codes are algebraic-geometric? *IEEE Transactions on Information Theory*, 37(3):583–602, 1991.

[8] M. A. Shokrollahi and H. Wasserman. List decoding of algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(2):432–437, 1999.

[9] H. Stichtenoth. *Algebraic Function Fields and Codes*. Springer-Verlag, 1993.

[10] M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13:180–193, 1997.

[11] M. Sudan. Decoding Reed-Solomon codes beyond the error-correction diameter. In *Proceedings of the 35th Annual Allerton Conference on Communication, Control and Computing*, 1997.

[12] M. A. Tsfasman and S. G. Vlăduţ. *Algebraic-Geometric Codes*. Mathematics and its Applications. Kluwer Academic Publishers, 1991.

[13] J. von zur Gathen and J. Gerhard. *Computer Algebra*. Cambridge University Press, 1999.