Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach

Magali Bardet, Rocco Mora, Jean-Pierre Tillich

Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach

Magali Bardet [*1,3], Rocco Mora [†2,3], and Jean-Pierre Tillich [‡3]

[1]LITIS, University of Rouen Normandie
[2]Sorbonne Universités, UPMC Univ Paris 06
[3]Inria, Team COSMIQ, 2 rue Simone Iff, CS 42112, 75589 Paris Cedex 12, France

Abstract

Decoding a Reed-Solomon code can be modeled by a bilinear system which can be solved by Gröbner basis techniques. We will show that in this particular case, these techniques are much more efficient than for generic bilinear systems with the same number of unknowns and equations (where these techniques have exponential complexity). Here we show that they are able to solve the problem in polynomial time up to the Sudan radius. Moreover, beyond this radius these techniques recover automatically polynomial identities that are at the heart of improvements of the power decoding approach for reaching the Johnson decoding radius. They also allow to derive new polynomial identities that can be used to derive new algebraic decoding algorithms for Reed-Solomon codes. We provide numerical evidence that this sometimes allows to correct efficiently slightly more errors than the Johnson radius.

1 Introduction

Decoding a large number of errors in Reed-Solomon codes. A long-standing open problem in algebraic coding theory was that of decoding Reed-Solomon codes beyond the error-correction radius, $\frac{1-R}{2}$ (where R stands for the code rate). This problem was solved in a breakthrough paper by Sudan in [Sud97] where it was shown that there exists an algebraic decoder that works up to a fraction of errors $1-\sqrt{2R}$ (the so called Sudan radius here). This was even improved later on by Guruswami and Sudan in [GS98] with a decoder that works up to the Johnson radius $1-\sqrt{R}$. This represents in a sense the limit for such decoders since these decoders are list decoders that output all codewords up to this radius and beyond this radius the list size is not guaranteed to be polynomial anymore. However, if we do not insist on having a decoder that outputs all codewords within a certain radius, or if we just want a decoder that is successful most of the time on the q-ary symmetric channel of crossover probability p, then we can still hope to have an efficient decoder beyond this bound. Moreover, it is even interesting to investigate if there are decoding algorithms of say subexponential complexity above the radius $1-\sqrt{R}$.

[*]magali.bardet@univ-rouen.fr
[†]rocco.mora@inria.fr
[‡]jean-pierre.tillich@inria.fr

A Gröbner basis approach. Our approach to this problem is to express the decoding problem as a bilinear system and explore what Gröbner bases techniques have to tell us in this case. Indeed consider a k-dimensional Reed-Solomon code of length n over $\mathbb{F}_q$ with support $\mathbf{a} = (a_i)_{1\leq i\leq n} \in \mathbb{F}_q^n$:

$$\mathbf{RS}_k(\mathbf{a}) = \{(P(a_i))_{1\leq i\leq n} : \ P \in \mathbb{F}_q[X], \ \deg(P) < k\}.$$

Let $\mathbf{b} = (b_i)_{1\leq i\leq n}$ be the received word,

$\mathcal{E}$ be the set of positions in error and define the error locator as usual

$$\Lambda(X) \stackrel{\text{def}}{=} \Pi_{i\in\mathcal{E}}(X - a_i). \tag{1}$$

From this, we can write the bilinear system with unknowns the coefficients p_i of the polynomial $P(X) = \sum_{i=0}^{k-1} p_i X^i$ corresponding to the codeword that was sent and the coefficients λ_j of the error locator polynomial $\Lambda(X) = X^t + \sum_{j=0}^{t-1} \lambda_j X^j$ if we assume that there were t errors. We have n bilinear equations in the p_i's and the λ_j's coming from the n relations $P(a_\ell)\Lambda(a_\ell) = b_\ell \Lambda(a_\ell)$, $\ell \in [\![1, n]\!]$, namely

$$\sum_{i=0}^{k-1}\sum_{j=0}^{t} a_\ell^{i+j} p_i \lambda_j = \sum_{j=0}^{t} b_\ell a_\ell^j \lambda_j, \ \ \ell \in [\![1, n]\!] \text{ and } \lambda_t = 1. \tag{2}$$

Gröbner basis techniques: a simple and automatic way for obtaining a polynomial time algorithm in our case. Standard Gröbner bases techniques can be used to solve this system, however solving (2) is much easier than solving a generic bilinear system. In particular these techniques solve typically in polynomial time the decoding problem when the fraction of errors is below the Sudan radius. This is explained in Section 3.1. The reason why the Gröbner basis approach works in polynomial time is related to power-decoding [SSB10, Nie14] and can be explained by similar arguments. However, the nice thing about this Gröbner basis approach is that the algorithm itself is very simple and can be given without any reference to power decoding (or the Sudan algorithm). The computation of the Gröbner basis reveals degree falls which are instrumental for its very low complexity. Understanding these degree falls can be explained by the polynomial equations used by power decoding. However, this simple algorithm also appears to be very powerful beyond the Sudan bound: experimentally it seems that it is efficient up to the Johnson radius and that it is even able to correct more errors in some cases than the refinement of the original power decoding algorithm [Nie18] (which reaches asymptotically the Johnson radius). This is demonstrated in Section 4.

Understanding the nice behavior of the Gröbner basis approach. Moreover, trying to understand theoretically why this algorithm behaves so well, is not only explained by the polynomial relations which are at the heart of the power decoding approach, it also reveals new polynomial relations that are not exploited by the power decoding approach as shown in Section 3. In other words, this approach not only gives an efficient algorithm, it also exploits other polynomial relations. It seems fruitful to understand and describe them, this namely paves the road towards new algebraic decoders of Reed-Solomon codes.

Notation. Throughout the paper we will use the following notation. The integer interval $\{a, a+1, \cdots, b\}$ will be denoted by $[\![a, b]\!]$. For a polynomial $Q(X) = \sum_{i=0}^{m} q_i X^i$, $\text{coeff}(Q(X), X^s)$ stands for the coefficient q_s of X^s in $Q(X)$. For two polynomials $Q(X)$ and $G(X)$, $[Q(X)]_{G(X)}$ stands for the remainder of $Q(X)$ divided by $G(X)$.

2 The Algorithm

Consider an algebraic system of equations

$$\begin{cases} f_1(x_1, \cdots, x_\ell) &= 0 \\ \cdots &= \\ f_m(x_1, \cdots, x_\ell) &= 0 \end{cases} \tag{3}$$

where the f_i's are polynomials in $x_1, \cdots, x_\ell$. Such systems can be solved by Gröbner basis techniques (see [CLO15] for instance). To simplify the discussion assume that we have a unique solution to the algebraic system (3) and that the polynomial ideal $\mathcal{I}$ generated by the f_i's is radical, meaning that whenever there is a polynomial f and a positive integer s such that f^s is in $\mathcal{I}$, then f is in $\mathcal{I}$. This seems to be the typical case for (2) when the number of errors is below the Gilbert-Varshamov bound. In such a case, the reduced Gröbner basis of the ideal $\mathcal{I}$ is given by the set $\{x_1 - r_1, \cdots, x_n - r_\ell\}$ for any admissible monomial ordering, where $(r_1, \cdots, r_\ell)$ stands for the unique solution of (3) (this is standard, see for instance [Bar04, Lemma 2.4.3, p.40]). Recall here that a Gröbner basis of a polynomial ideal is defined for a given admissible monomial ordering[1] as a generating set $\{g_1, \cdots, g_s\}$ of the ideal such that the ideal generated by the leading monomials $LM(g_i)$ (where $LM(g)$ is the largest monomial in g) of the g_i's coincides with the ideal generated by all the leading monomials of the elements of $\mathcal{I}$:

$$\langle LM(g_1), \cdots, LM(g_s) \rangle = \langle LM(f) : f \in \mathcal{I} \rangle.$$

We will adopt Lazard's point of view [Laz83] to compute a Gröbner basis and use Gaussian elimination on the Macaulay matrices associated to the system. The main known and efficient algorithm for this is Faugère's F4 algorithm [Fau99], see [CLO15] for background on the subject.

We recall that the Macaulay matrix $\mathcal{M}_D^{\text{acaulay}}(\mathcal{S})$ in degree D of a set $\mathcal{S} = \{f_1, \cdots, f_m\}$ of polynomials is the matrix whose columns correspond to the monomials of degree $\leq D$ sorted in descending order w.r.t. a chosen monomial ordering, whose rows correspond to the polynomials tf_i for all i where t is a monomial of degree $\leq D - \deg(f_i)$, and whose entry in row tf_i and column u is the coefficient of the monomial u in the polynomial tf_i. A Gröbner basis for the system can be computed by computing a row echelon form of $\mathcal{M}_D^{\text{acaulay}}$ for large enough D [Laz83] and [CLO15, chap. 10]. However, this way of solving (2) is very inefficient (unless $t \leq \frac{n-k}{2}$ where direct row echelonizing (2) is enough) because during the Gaussian elimination process we have a sequence of degree falls which are instrumental for computing a Gröbner basis by staying at a very small degree (this appears clearly if we use for instance Faugère's F4 algorithm [Fau99] on (2)).

A <u>degree fall</u> is a polynomial combination $\sum_{i=1}^m g_i f_i$ of the f_i's which satisfies

$$0 < s \stackrel{\text{def}}{=} \deg\left(\sum_{i=1}^m g_i f_i\right) < \max_{i=1}^m \deg(g_i f_i).$$

We say that $\deg(\sum_{i=1}^m g_i f_i)$ is a degree fall of <u>degree</u> s.

The simplest example of such a degree fall occurs in (2) when $t < n - k$. Here there are linear combinations of the bilinear equations of (2) giving linear equations. This can be

[1] This is a total ordering $<$ of the monomials such that (i) $m < m' \implies mt < m't$ for any monomial t (ii) every subset of monomials has a smallest element.

verified by performing the change of variables $z_s \stackrel{\text{def}}{=} \sum_{i,j:i+j=s} p_i\lambda_j$ in (2) and get the system

$$\sum_{s=0}^{t+k-1} a_\ell^s z_s = \sum_{j=0}^{t} b_\ell a_\ell^j \lambda_j, \quad \ell \in [\![1, n]\!]. \tag{4}$$

In other words, by eliminating the z_s's in these equations we obtain linear equations involving only the λ_i's. When $t \leq \frac{n-k}{2}$ there are enough such equations to recover from them the λ_i's and by substituting for them in (2) the p_i's by solving again a linear system. Of course, this is well known, and there are much more efficient algorithms for solving this system but still it is interesting to notice that the Gröbner basis approach already yields a polynomial time algorithm for the particular bilinear system (2) despite being exponential (for a large range of parameters) for generic bilinear systems with the same number of unknowns and equations as (2) [FSS11, Spa12].

A slightly less trivial degree fall behavior is obtained in the case the fraction of errors is the Sudan radius. Here, after substituting for the λ_i's which can be expressed as linear functions of the other λ_i's by using the aforementioned linear equations involving the λ_i's we obtain new bilinear equations $f'_1, \cdots, f'_m$. It turns out that we can perform linear combinations on these f'_i's to eliminate the monomials of degree 2 in them and derive new linear equations involving only the λ_i's. This is proved in Subsection 3.1. This process can be iterated and there are typically enough such linear equations to recover the λ_i's in this way as long as t is below or equal to the Sudan decoding radius. As explained above, this allows to recover the right codeword by plugging the values for λ_i in (2) and solving the corresponding linear system in the p_i's.

Note that here, and in all the paper, we are considering graded monomial orderings (a monomial of degree d is always smaller than a monomial of degree $d' > d$). Through this paper, we use the notion of affine D-Gröbner basis, which is the truncated Gröbner basis obtained by ignoring computations in degree greater than D. It is well known that there exists a D such that a D-Gröbner basis is indeed a Gröbner basis. We describe here Algorithm 1 which computes a D-Gröner basis of a given system through linear algebra. It is less efficient than standard algorithms but has the merit of being simple and showing what is computed during such algorithms. It is also of polynomial time complexity when D is fixed. It uses the function $\texttt{Pol}(M)$ that returns the polynomials represented by the rows of a Macaulay matrix M.

Algorithm 1 D-Gröbner Basis

Input
D Maximal degree,
$\mathcal{S}$ $= \{f_1, \cdots, f_m\}$ set of polynomials.

repeat
$\mathcal{S} \leftarrow \texttt{Pol}(\texttt{EchelonForm}(\mathcal{M}_D^{\text{acaulay}}(\mathcal{S})))$
until $\dim_{\mathbb{F}_q} \mathcal{S}$ has not increased.
Output $\mathcal{S}$.

It is clear that Algorithm 1 terminates and has a polynomial complexity if D is fixed. The previous remarks show that it is possible to decode up to the Sudan decoding radius with $D = 2$. However, when the number of errors becomes bigger, $D = 2$ is not enough to exhibit more degree falls. We have to go to a higher degree. It turns out that already taking $D = 3$

yields very interesting degree falls that are instrumental to the generalization of the power decoding approach of [Nie18] decoding up to the Johnson radius.

3 A partial explanation of the algebraic behavior

3.1 Correcting up to the Sudan bound in polynomial time

The efficiency of Algorithm 1 is already demonstrated by the fact that choosing $D = 2$ in it corrects in polynomial time as many errors as Sudan's algorithm. Roughly speaking the choice of $D = 2$ in Algorithm 1 means that we just keep the equations of degree 2 and try to produce new linear equations by linear combinations of the equations of degree 2 aiming at eliminating the degree 2 monomials. The reason why this algorithm is so efficient is related to power decoding [SSB10]: the algorithm finds automatically the linear equations exploited by the power decoding approach. It is here convenient in order to explain the effectiveness of the Gröbner basis approach to bring in an equivalent algebraic system which is basically the key equation implicit in Gao's decoder [Gao03] (and the one used in the power decoding approach) which is the following polynomial equation:

$$P(X)\Lambda(X) \equiv R(X)\Lambda(X) \mod G(X) \tag{5}$$

where $R(X)$ is the polynomial of degree $\leq n-1$ interpolating the received values, i.e

$$R(a_\ell) = b_\ell, \;\; \ell \in [\![1, n]\!] \text{ and } G(X) \stackrel{\text{def}}{=} \Pi_{\ell=1}^{n}(X - a_\ell).$$

Note that these two polynomials are immediately computable by the receiver (and G can be precomputed). By using the same unknowns as in (2), namely the coefficients of $P(X)$ and $\Lambda(X)$ we obtain a bilinear system with n equations. It is readily seen that

Proposition 1. *The bilinear systems* (2) *and* (5) *are equivalent:* (5) *can be obtained from linear combinations of* (2) *and vice versa.*

This follows on the spot from

Fact 2. *For any polynomial $Q(X) \in \mathbb{F}_q[X]$ of degree $< n$, the coefficients of Q can be expressed as linear combinations of $Q(a_1), \cdots, Q(a_n)$.*

This fact is just a consequence that Q coincides with its interpolation polynomial on the points $(a_\ell, Q(a_\ell))$ and that this interpolation polynomial is given by

$$Q(X) = \sum_{\ell=1}^{n} Q(a_\ell) \frac{\Pi_{j\neq\ell}(X - a_j)}{\Pi_{j\neq\ell}(a_\ell - a_j)}.$$

To understand now why (5) can be derived from (2), we just notice that if we bring in

$$\begin{aligned} Q(X) &\stackrel{\text{def}}{=} P(X)\Lambda(X) - R(X)\Lambda(X) \\ S(X) &\stackrel{\text{def}}{=} Q(X) \mod G(X), \end{aligned}$$

then

- (2) amounts to write $Q(a_\ell) = 0$ for ℓ in $[\![1, n]\!]$ and to express the $Q(a_\ell)$'s as quadratic forms in the λ_i's and the p_j's.
- Since $Q(a_\ell) = S(a_\ell)$ for all ℓ in $[\![1, n]\!]$ and since S is of degree $< n$ we can use the previous fact and express its coefficients linearly in terms of the $S(a_\ell) = Q(a_\ell)$'s.
- Since (5) is nothing but expressing that the coefficients of $S(X)$ are all equal to 0, we obtain that the equations of (5) can be obtained from linear combinations of the equations of (2).

Conversely since $S(a_\ell)$ can be written as a linear combination of the coefficients of $S(X)$, the quadratic equations in the λ_i's and the p_i's obtained by writing $S(a_\ell) = 0$ are linear combinations of the quadratic equations given by (5). These equations $S(a_\ell) = 0$ coincide with the equations in (2), since $Q(a_\ell) = S(a_\ell)$ for all ℓ in $[\![1, n]\!]$.

The point of (5) is that

- These equations are more convenient to work with to understand what is going on algebraically during the Gröbner basis calculations of Algorithm 1.
- They give directly $n - k - t + 1$ linear equations, since (i) the coefficient of $S(X)$ of degree $d \in [\![t+k, n-1]\!]$ coincides with the coefficient of the same degree in $-R(X)\Lambda(X)$ mod $G(X)$ since $\Lambda(X)P(X)$ is of degree $\leq t+k-1$; (ii) the coefficient of $S(X)$ of degree $t + k - 1$ is equal to $p_{k-1} - \text{coeff}\left([\Lambda(X)R(X)]_{G(X)}, X^{t+k-1}\right)$ because $\Lambda(X)$ is monic and of degree t.

With this at hand we can now prove that

Proposition 3. *Let* $q_1 \stackrel{def}{=} \max\{u : t+(k-1)u \leq n-1\} = \left\lfloor \frac{n-t-1}{k-1} \right\rfloor$. *All affine functions in the* λ_i*'s of the form* $\text{coeff}\left(\left[\Lambda(X)R^j(X)\right]_{G(X)}, X^u\right)$ *for* $j \in [\![1, q_1]\!]$ *and* $u \in [\![t+(k-1)j+1, n-1]\!]$ *are in the linear span of the set* $\mathcal{S}$ *output by Algorithm 1 when* $D = 2$.

Remark 4. *The fact that these are indeed affine functions follows on the spot from generalizing the degree considerations above:* $\Lambda(X)P(X)^j$ *is of degree* $\leq t + (k-1)j$.

Proof. Notice that from the equivalence we have just proved, the space generated by $\mathcal{S}$ contains initially (and therefore all the time) the space of affine functions in the λ_i's generated by

$$\text{coeff}\left([-\Lambda(X)R(X)]_{G(X)}, X^u\right) = \text{coeff}\left([\Lambda(X)P(X) - \Lambda(X)R(X)]_{G(X)}, X^u\right), \quad \text{for all } u \in [\![t+k, n-1]\!].$$

Now proceed by induction on j, and assume that at some point the space generated by $\mathcal{S}$ contains the linear span of the affine functions

$$\text{coeff}\left(\left[-\Lambda(X)R^j(X)\right]_{G(X)}, X^u\right) = \text{coeff}\left(\left[\Lambda(X)P(X)^j - \Lambda(X)R(X)^j\right]_{G(X)}, X^u\right),$$

for all $u \in [\![t + (k-1)j + 1, n-1]\!]$ where j is some integer in the interval $[\![1, q_1 - 1]\!]$. Note that

$$\begin{aligned} &\left(\Lambda P^{j+1} - \Lambda R^{j+1}\right) \mod G \qquad (6)\\ &= \left(P(\Lambda P^j - \Lambda R^j) + R^j(\Lambda P - \Lambda R)\right) \mod G \\ &= \left(P(\Lambda P^j - \Lambda R^j \mod G) + R^j(\Lambda P - \Lambda R \mod G)\right) \mod G. \qquad (7)\end{aligned}$$

We use the equality between the polynomials (6) and (7) to claim that their coefficients should coincide for all the degrees $[\![t+(j-1)(k-1), n-1]\!]$. Note now that after the elimination of variables performed so far, this makes that all coefficients of degree in $[\![t+(k-1)j+1, n-1]\!]$ in $\Lambda P^j - \Lambda R^j \mod G$ vanish, since they were affine functions by the induction hypothesis and become 0 after the variable elimination step. This implies that $\Lambda P^j - \Lambda R^j \mod G$ becomes a polynomial of degree $\leq t+(k-1)j$ after elimination of variables. Therefore $P(\Lambda P^j - \Lambda R^j \mod G)$ is a polynomial of degree $\leq t+(k-1)(j+1)$. From the equality of the polynomials (6) and (7), this implies that the coefficient of degree u in $\left(\Lambda P^{j+1} - \Lambda R^{j+1}\right) \mod G$ coincides with the coefficient of the same degree in $(R^j(\Lambda P - \Lambda R \mod G)) \mod G$ for u in $[\![t+(k-1)(j+1)+1, n-1]\!]$. We observe now that the last coefficient is nothing but a linear combination of the coefficients of $\Lambda P - \Lambda R \mod G$, which are precisely the initial polynomial equations. Since the polynomial $\left(\Lambda P^{j+1} - \Lambda R^{j+1}\right) \mod G$ has all its coefficients that are affine functions in the λ_i's by Remark 4 for all the degrees $u \in [\![t+(k-1)(j+1)+1, n-1]\!]$ we obtain that after the Gaussian elimination step, $\mathcal{S}$ contains the space generated by these aforementioned affine functions. This proves the proposition by induction on j. □

These linear equations that we produce coincide exactly with the linear equations produced by the power decoding approach [SSB10] and this allows to correct as many errors as the power decoding approach based on the same assumption, namely that they are all independent, which is actually the typical scenario. However, contrarily to power decoding that is bound to make such an assumption to work, the Gröbner basis is more versatile, as it allows to decode even without this assumption as explained in Section 4.

3.2 Decoding up to the Johnson radius

The power decoding approach of [SSB10] was generalized in [Nie18] to decode up to the Johnson radius by bringing in the "error evaluator" polynomial $\Omega(X)$ of degree $\leq t-1$ defined by

$$\Omega(a_i) = -e_i, \quad \text{for all } i \in [\![1, n]\!] \text{ for which } e_i \neq 0. \tag{8}$$

where e_i is the error value at position i. In other words, it is the interpolation polynomial defined by (8) for all i that are in error. This approach then crucially relies on the following identity ([Nie18, Lemma 2.1]):

$$\Lambda(P-R) = \Omega G. \tag{9}$$

The generalization of power decoding then uses this identity to derive further identities that are summarized by the following formulas (this is Theorem 3.1 in [Nie18]), for any positive integer s and v such that $s \leq v$ we have

$$\Lambda^s P^u = \sum_{i=0}^{u} \left(\Lambda^{s-i}\Omega^i\right) \binom{u}{i} R^{u-i} G^i \qquad u \in [\![1, s-1]\!], \tag{10}$$

$$\Lambda^s P^u \equiv \sum_{i=0}^{s-1} \left(\Lambda^{s-i}\Omega^i\right) \binom{u}{i} R^{u-i} G^i \mod G^s \qquad u \in [\![s, v]\!]. \tag{11}$$

The approach of [Nie18] relies on the fact that when the number of errors is below the Johnson radius, there is a choice of s and v such that the total number of coefficients of the polynomials Λ^s, $\Lambda^{s-1}\Omega$, $\cdots$, Ω^s as well as $\Lambda^s P$,$\cdots$, $\Lambda^s P^v$ is less than or equal to the number of equations linking these coefficients coming from (10) and (11). In this case (and if these equations are

independent) we recover them by solving the corresponding linear system. Notice that with this strategy, there is for a given value s a maximal value for u given by

$$q_s \stackrel{\text{def}}{=} \max\{u : st + u(k-1) \leq sn - 1\}.$$

It is readily seen that taking larger of u only increase the number of variables in the linear system without being able to make it determinate if it was not determinate before. Interestingly enough our Gröbner basis approach also exhibits degree falls of degree s that are related to the equations (10) and (11). This can be understood by using an equivalent definition of Ω which is better suited to understand our Gröbner basis approach. We namely define Ω as

$$\Omega \stackrel{\text{def}}{=} -\Lambda R \div G. \tag{12}$$

Notice that from this definition we directly derive two results

1. The coefficients of Ω are affine functions of the λ_i's.
2. As long as $t \leq n - k$, (9) follows from (12) and (5). Indeed $[\Lambda R]_G = \Lambda P$. This follows from (5) and $t + k - 1 \leq n - 1$ implying that $\Lambda P = \Lambda R \mod G$. This and (12) then imply that $\Lambda R = -\Omega G + \Lambda P$ which is obviously equivalent to (9).

Note that from these considerations, that if we equate the coefficients of the polynomials in (10) for all the degrees in $[\![st + u(k-1) + 1, st + u(n-1)]\!]$ and in (11) for all the degrees in $[\![st + u(k-1) + 1, s(n-1)]\!]$, the coefficient of the left-hand term vanishes and the coefficient in the righthand term is a polynomial of degree s in the λ_i's (this follows from the fact that the coefficients of Ω are affine functions in those λ_i's). This gives polynomial equations in the λ_i's of degree s. In a sense, they can be viewed as generalizations at degree s of the linear equations that were mentioned when Algorithm 1 is applied when $D = 2$. These equations are actually produced as degree falls that are in the linear span of intermediate sets $\mathcal{S}$ produced in Algorithm 1 when $D = s + 1$. There are also other equations of degree s produced by Algorithm 1 in such a case. To explain this point it makes sense to bring in notation for the right-hand term in (10) and (11). Let us define

$$\begin{aligned}
\chi(s,u) &\stackrel{\text{def}}{=} \sum_{i=0}^{u} \binom{u}{i} \Lambda^{s-i} R^{u-i} \Omega^i G^i = \Lambda^{s-u} \left(\Lambda R + \Omega G\right)^u \quad \text{if } u < s, \\
\chi(s,u) &\stackrel{\text{def}}{=} \left[\sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-i} R^{u-i} \Omega^i G^i\right]_{G^s} \quad \text{if } u \geq s
\end{aligned}$$

We also let $\chi(s,u)_H$ be the polynomial where we dropped all the terms of degree $\leq ts + u(k-1)$ in $\chi(s,u)$, i.e. $\chi(s,u) = \sum_i a_i X^i$, then $\chi(s,u)_H = \sum_{i > ts+u(k-1)} a_i X^i$.

Theorem 5. *Let $\mathcal{I}_D = \langle \mathcal{S} \rangle_{\mathbb{F}_q}$ where $\mathcal{S}$ is the set output by Algorithm 1. We have for all nonnegative integers s, s', $u \leq q_s$, $u' \leq q_{s'}$*

$$\chi(s,u)_H \quad \in_{coef} \quad \mathcal{I}_{s+1} \tag{13}$$

$$\chi(s,u)\chi(s',u') - \chi(s+s',u+u') \quad \in_{coef} \quad \mathcal{I}_{s+s'+1}. \tag{14}$$

where $P \in_{coef} \mathcal{I}_v$ (where P is a polynomial with coefficients that are polynomials in the λ_i's and the p_i's) means that all the coefficients of P belong to $\mathcal{I}_v$.

From (10) and (11) it is of course clear that $\chi(s,u)\chi(s',u') - \chi(s+s',u+u')$ belongs to the ideal generated by the polynomial equations since they basically come from the identity $\Lambda^s P^u \Lambda^{s'} P^{u'} = \Lambda^{s+s'} P^{u+u'}$. What is somehow surprising is that these equations are actually discovered at a rather small degree Gröbner basis computation (namely by staying at degree $s+s'+1$). Moreover these equations only involve the λ_i's. By inspection of the behavior of the Gröbner basis computation, it seems that the linear equations that we produce later on are first produced by degree falls only involving these equations of degree s. It is therefore tempting to change the Gröbner basis decoding procedure strategy: instead of feeding Algorithm 1 with the initial system (2) or (5) we run it with the equations of degree s given by Theorem 5. Once we have recovered the λ_i's in this way we recover the p_i's by solving a linear system as explained earlier. How this strategy behaves on non-trivial examples is now explained in the next section.

This result is proved in the following subsection.

3.3 Proof of Theorem 5

It will be convenient here to notice that $\chi(s,s)$ has a slightly simpler expression which avoids the reduction modulo G^s.

Lemma 6.

$$\chi(s,s) = (\Lambda R + \Omega G)^s.$$

Proof. $\chi(s,s)$ is defined as

$$\begin{aligned}
\chi(s,s) &\stackrel{\text{def}}{=} \left[\sum_{i=0}^{s-1} \binom{s}{i} \Lambda^{s-i} R^{s-i} \Omega^i G^i\right]_{G^s} \\
&= \left[\sum_{i=0}^{s} \binom{s}{i} \Lambda^{s-i} R^{s-i} \Omega^i G^i\right]_{G^s} \\
&= [(\Lambda R + \Omega G)^s]_{G^s} \\
&= (\Lambda R + \Omega G)^s
\end{aligned}$$

□

It will also be helpful to observe that $\chi(s,u)$ and $\chi(s,u+1)$ are related by the following identity

Lemma 7.

$$\begin{aligned}
\chi(s,u)P - \chi(s,u+1) &= \Lambda^{s-u-1}(\Lambda R + \Omega G)^u (\Lambda P - \Lambda R - \Omega G) \quad \textit{for } u \in [\![0, s-1]\!] \\
[\chi(s,u)P - \chi(s,u+1)]_{G^s} &= \left[(\Lambda P - \Lambda R - \Omega G) \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i\right]_{G^s} \quad \textit{for } u \in [\![s, q_s - 1]\!].
\end{aligned}$$

Proof. For $u \in [\![0, s-1]\!]$ we have (for the case $u = s-1$ we use Lemma 6 for the term $\chi(s,u+1)$):

$$\begin{aligned}
\chi(s,u)P - \chi(s,u+1) &= \Lambda^{s-u} P (\Lambda R + \Omega G)^u - \Lambda^{s-u-1} P (\Lambda R + \Omega G)^{u+1} \\
&= \Lambda^{s-u-1}(\Lambda R + \Omega G)^u (\Lambda P - \Lambda R - \Omega G).
\end{aligned}$$

For $u \in [\![s, q_s - 1]\!]$ we observe that

$$\begin{aligned}
[\chi(s,u)P]_{G^s} &= \left[P \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-i} R^{u-i} \Omega^i G^i\right]_{G^s} \\
&= \left[\Lambda P \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i\right]_{G^s} \qquad (15)
\end{aligned}$$

and

$$\begin{aligned}
(\Lambda R + \Omega G) \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i &= \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-i} R^{u+1-i} \Omega^i G^i + \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^{i+1} G^{i+1} \\
&= \Lambda^s R^{u+1} + \binom{u}{s-1} R^{u-s+1} \Omega^s G^s \\
&\quad + \sum_{i=1}^{s-1} \left(\binom{u}{i} + \binom{u}{i-1} \right) \Lambda^{s-i} R^{u+1-i} \Omega^i G^i \\
&= \binom{u}{s-1} R^{u-s+1} \Omega^s G^s + \sum_{i=0}^{s-1} \binom{u+1}{i} \Lambda^{s-i} R^{u+1-i} \Omega^i G^i
\end{aligned}$$

This implies

$$\chi(s, u+1) = \left[(\Lambda R + \Omega G) \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i\right]_{G^s}. \qquad (16)$$

The second equation of the lemma follows directly from (15) and (16). □

A last lemma will be helpful now

Lemma 8. *For all nonnegative integers s and $u < q_s$*

$$\chi(s,u)P - \chi(s,u+1) \in_{coef} \mathcal{I}_{s+1} \qquad (17)$$

$$\chi(s,u+1)_H \in_{coef} \mathcal{I}_{s+1}. \qquad (18)$$

Proof. We will prove this lemma by induction on u. For $u \leq s-1$ we observe from Lemma 7 that

$$\begin{aligned}
\chi(s,u)P - \chi(s,u+1) &= \Lambda^{s-u-1} (\Lambda R + \Omega G)^u (\Lambda P - \Lambda R - \Omega G) \qquad (19) \\
&\in_{\text{coef}} \mathcal{I}_{s+1}
\end{aligned}$$

The last point follows from the fact that (19) implies that the coefficients of $\chi(s,u)P - \chi(s,u+1)$ are clearly in the space spanned by $\mathcal{S}$ once we multiply the original f_i's (i.e. the coefficients of $\Lambda P - \Lambda R - \Omega G$) by all monomials of degree $\leq s-1$) because the coefficients of $\Lambda^{s-u-1}(\Lambda R + \Omega G)^u$ are polynomials of degree $\leq s-1$ in the λ_i's.

This also implies that $\chi(s,u+1)_H \in_{\text{coef}} \mathcal{I}_{s+1}$, since $\deg(\chi(s,u)P) = ts + u(k-1)$. Now let us assume that $\chi(s,u-1)P - \chi(s,u) \in_{\text{coef}} \mathcal{I}_{s+1}$ and $\chi(s,u)_H \in_{\text{coef}} \mathcal{I}_{s+1}$, for some $s \leq u < q_s$. From Lemma 7 we know that

$$[\chi(s,u)P - \chi(s,u+1)]_{G^s} = \left[(\Lambda P - \Lambda R - \Omega G) \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i\right]_{G^s}.$$

Therefore

$$[\chi(s,u)P - \chi(s,u+1)]_{G^s} \in_{\text{coef}} \mathcal{I}_{s+1}$$

since clearly (i)

$$(\Lambda P - \Lambda R - \Omega G)\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^i G^i \in_{\text{coef}} \mathcal{I}_{s+1}$$

$(\Lambda P - \Lambda R - \Omega G)\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^i G^i$.

By the induction hypothesis $\chi(s,u)_H \in_{\text{coef}} \mathcal{I}_{s+1}$ and such coefficients have degree s, then the coefficients corresponding to degrees $> ts + (u+1)(k-1)$ of $\chi(s,u)P$ belong to $\mathcal{I}_{s+1}$ too. Since $[\chi(s,u)P - \chi(s,u+1)]_{G^s} = [\chi(s,u)P]_{G^s} - \chi(s,u+1)$, it follows that

$$\chi(s,u)P - \chi(s,u+1) \in \mathcal{I}_{s+1}.$$

Thus, we also have $\chi(s,u+1)_H \in \mathcal{I}_{s+1}$. □

We are ready now to prove Theorem 5.

Proof of Theorem 5. We proceed by induction on u_1 and u_2. We first observe that we trivially have $\chi(s_1,0)\chi(s_2,0) - \chi(s_1+s_2,0) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1}$ since

$$\chi(s_1,0)\chi(s_2,0) - \chi(s_1+s_2,0) = \Lambda^{s_1}\Lambda^{s_2} - \Lambda^{s_1+s_2} = 0.$$

Now assume that we have

$$\chi(s_1,u_1)\chi(s_2,u_2) - \chi(s_1+s_2,u_1+u_2) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1},$$

for some positive integers s_1 and s_2 and non-negative integers $u_1 < q_{s_1}$ and $u_2 \leq q_{s_2}$. Since $\chi(s_1,u_1)\chi(s_2,u_2)$ and $\chi(s_1+s_2,u_1+u_2)$ are polynomials where all coefficients are polynomials in the λ_i's of degree $\leq s_1+s_2$, we also have

$$P\left(\chi(s_1,u_1)\chi(s_2,u_2) - \chi(s_1+s_2,u_1+u_2)\right) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1}. \tag{20}$$

By Lemma 8 we know that

$$P\chi(s_1,u_1) - \chi(s_1,u_1+1) \in_{\text{coef}} \mathcal{I}_{s_1+1}.$$

This implies

$$P\chi(s_1,u_1)\chi(s_2,u_2) - \chi(s_1,u_1+1)\chi(s_2,u_2) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1}. \tag{21}$$

On the other hand, still by Lemma 8, we have

$$P\chi(s_1+s_2,u_1+u_2) - \chi(s_1+s_2,u_1+u_2+1) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1}. \tag{22}$$

From (21) and (22) we derive that

$$-P\chi(s_1,u_1)\chi(s_2,u_2)+\chi(s_1,u_1+1)\chi(s_2,u_2)+P\chi(s_1+s_2,u_1+u_2)-\chi(s_1+s_2,u_1+u_2+1) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1} \tag{23}$$

(23) and (20) imply that

$$\chi(s_1,u_1+1)\chi(s_2,u_2) - \chi(s_1+s_2,u_1+u_2+1) \in_{\text{coef}} \mathcal{I}_{s_1+s_2+1}.$$

This proves the theorem by induction (the induction on u_2 follows directly from the fact we can exchange the role of u_1 and u_2). □

4 Experimental Results

In this section, we compare the behavior of a D-Gröbner basis computation on the bilinear system (5), with a system involving equations in the λ_j's only. We give examples where Johnson's bound is attained and passed.

The systems in λ_j's we use contains equations $\chi(s,u)_H$ and some relations $\chi(s,u)\chi(s',u') - \chi(s+s',u+u')$. Experimentally, they are linearly dependent from $\chi(s+s'-1,u+u')\chi(1,0) - \chi(s+s',u+u')$ and $\chi(s,q_s)_H$. Moreover, $\chi(s-1,u)\chi(1,0) \mod G^{s-1} = \chi(s,u) \mod G^{s-1}$, so we will consider equations $\mathcal{M}_{s,u}$ defined by

$$(\chi(s-1,u)\chi(1,0) - \chi(s,u)) \div G^{s-1}. \qquad (\mathcal{M}_{s,u})$$

We do not add equations that are polynomially dependent from $\chi(s,q_s)_H$ or $\mathcal{M}_{s+1,q_s}$ at degree at most D, and thus unnecessary for the computation.

Tables 1, 2 and 3 show results for $[n,k]_q$ taking values $[64,27]_{64}$, $[256,63]_{256}$ and $[37,5]_{61}$. The column $\#\lambda_j$ indicates the number of remaining λ_j's after elimination of the linear ones from the $\chi(1,*)_H$ relations. The column "Eq" indicates the equations used. The column "#Eq" contains the degrees of the equations[2].

We do our experiments using the `GroebnerBasis(S,D)` function in the computer algebra system `magma v2.25-6`. The practical complexity $\mathbb{C}$ is given by the `magma` function `ClockCycles`. For instance, on our machine with an Intel® Xeon® 2.00GHz processor, $2^{30.9}$ clock cycles are done in 1 second, $2^{36.8}$ in 1 minute and $2^{42.7}$ in 1 hour. "Max Matrix" indicates the size of the largest matrix during the process. The complexities include the computation of the equations $\chi(i,j)_H$ and $\mathcal{M}_{i,j}$ that could be improved.

For systems where the number of remaining λ_j's is small compared to the number of p_i's, e.g. Table 1 or Table 2, it is clearly interesting to compute a Gröbner basis for a system containing only polynomials in λ_j's: even if the maximal degree D is larger than for the bilinear system, the number of variables is much smaller and the computation is faster. For instance for $[n,k]_q = [64,27]_{64}$ in Table 1, on Johnson bound $t = 23$ the Gröbner basis for the bilinear system requires more than 6 hours of computation and 47 GB of memory, whereas the computation in λ_j's only takes less than a second. For $t = 24$ we couldn't solve the bilinear system directly, whereas the system in λ_j's only solves in less than a minute.

Table 2 gives an example where the number of λ_j's variables is quite large, but still smaller than the number of p_i's. The benefit of using equations in λ_j's only is clear.

On the contrary, Table 3 shows that for a small value of k compared to the number of λ_j's, the maximal degree for the bilinear system is smaller than the one for a system involving only λ_j's, but the total number of variables is almost the same, hence it is more interesting to solve directly the bilinear system. Moreover, here computing the $\mathcal{M}_{i,j}$ equations (that are equations in λ_j's of degree i) takes time. Note that, for $t \geq 26$ we may have several solutions: the Gröbner basis computation performs a list decoding and returns all the solutions.

5 Concluding remarks

This paper demonstrates that using a standard Gröbner basis computation on the bilinear system (5) for decoding a Reed-Solomon code is of polynomial complexity below Sudan's radius. The Gröbner basis computation reveals polynomial equations of small degree involving

[2] 2:45 means that the system contains 45 equations of degree 2.

Table 1: Experimental results for a $[n,k]_q = [64,27]_{64}$ RS-code. System (5) contains 26 variables p_i. Johnson's bound is $t = 23$.

t	$\#\lambda_j$	Eq.	#Eq.	D	Max Matrix	$\mathbb{C}$
19	1	(5)	2:45	2	65×57	$2^{22.2}$
		$\chi(2,3)_H$	2:11	2	45×28	$2^{23.7}$
20	3	(5)	2:46	3	1522×1800	$2^{26.5}$
		$\chi(2,3)_H$	2:9	2	47×28	$2^{24.4}$
21	5	(5)	2:47	3	1711×2889	$2^{27.1}$
		$\chi(2,3)_H + \chi(3,4)_H$	2:7, 3:24	3	66×56	$2^{26.8}$
22	7	(5)	2:48	4	31348×35972	$2^{36.1}$
		$\chi(2,3)_H + \chi(3,4)_H$	2:5, 3:21	4	271×283	$2^{27.6}$
23	9	(5)	2:49	5	428533×406773	$2^{45.4}$
		$\chi(2,3)_H + \mathcal{M}_{3,3}$	2:4, 3:22	5	1466×1641	$2^{30.1}$
24	11	(5)	2:50	≥ 6	–	–
		$\mathcal{M}_{3,3}$	2:1, 3:23	7	28199×23536	$2^{35.8}$

Table 2: Experimental results for a $[n,k]_q = [256,63]_{256}$ RS-code. System (5) contains 62 variables p_i. Johnson's bound is $t = 130$.

t	$\#\lambda_j$	Eq.	#Eq.	D	Max Matrix	$\mathbb{C}$
120	36	(5)	2:182	3	20023×128018	$2^{38.0}$
		$\chi(2,3)_H$	2:85	2	119×703	$2^{34.5}$
121	39	(5)	2:183	3	21009×143741	$2^{38.9}$
		$\mathcal{M}_{2,2}$	2:111	3	9780×8517	$2^{35.0}$
122	42	(5)	2:184	3	22050×160434	$2^{39.7}$
		$\mathcal{M}_{2,2}$	2:113	3	4858×14189	$2^{35.3}$
123	45	(5)	2:185	3	23112×178090	$2^{40.1}$
		$\mathcal{M}_{2,2}$	2:115	3	5289×17295	$2^{35.8}$
124	48	(5)	2:186	≥ 4	–	–
		$\mathcal{M}_{2,2} + \mathcal{M}_{4,6}$	2:117, 3:1, 4:189	4	164600×270725	$2^{45.2}$

Table 3: Experimental results for a $[n,k]_q = [37,5]_{61}$ RS-code. System (5) contains 4 variables p_i. Johnson's bound is $t = 24$, Gilbert-Varshamov's bound is $t = 28$.

t	$\#\lambda_j$	Eq.	#Eq.	D	Max Matrix	$\mathbb{C}$
24	12	(5)	2:28	3	1065×1034	$2^{26.0}$
		$\mathcal{M}_{2,3}$	2:37	3	454×454	$2^{28.0}$
25	15	(5)	2:29	3	2520×1573	$2^{28.0}$
		$\chi(2,5)_H +$ $\chi(3,8)_H +$ $\mathcal{M}_{2,2} + \mathcal{M}_{3,5}$	2:25, 3:40	4	3193×3311	$2^{34.3}$
26	18	(5)	2:30	4	20446×15171	$2^{33.1}$
		$\chi(2,5)_H +$ $\mathcal{M}_{2,2} + \mathcal{M}_{3,5}$ $+ \mathcal{M}_{4,8}$	2:25, 3:37, 4:37	5	38796×22263	$2^{38.1}$
27	21	(5)	2:31	4	27366×24894	$2^{36.0}$

the coefficients λ_i of the error locator polynomial. They are related to the power decoding approach [Nie18]. We give a theorem explaining why these polynomial relations are obtained at a surprisingly small degree. This is a first step for understanding why the Gröbner works surprisingly well beyond the Sudan radius and is successful by staying at a small degree. We have also explored another way of using this approach, namely by feeding some of the aforementioned polynomial relations directly in a Gröbner basis computation. This results in some cases in a considerable complexity gain. We have considered some of the examples given in [Nie18] and show that this Gröbner basis approach can still be effective slightly beyond Johnson's bound. We also remark that contrarily to the power decoding approach which is restricted to the case where there is a unique solution to the decoding problem, the Gröbner basis computation is also able to compute all solutions. This approach opens new roads for decoding algebraically a Reed-Solomon code.

References

[Bar04] Magali Bardet. Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. PhD thesis, Université Paris VI, December 2004. http://tel.archives-ouvertes.fr/tel-00449609/en/.

[CLO15] David Cox, John Little, and Donal O'Shea. Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra. Undergraduate Texts in Mathematics, Springer-Verlag, New York., 2015.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). J. Pure Appl. Algebra, 139(1-3):61–88, 1999.

[FSS11] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spacodecrenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity. J. Symbolic Comput., 46(4):406–437, 2011.

[Gao03] Shuhong Gao. A new algorithm for decoding Reed-Solomon codes. In Vijay K. Bhargava, H. Vincent Poor, Vahid Tarokh, and Seokho Yoon, editors, Communications, Information and Network Security, pages 55–68, Boston, MA, 2003. Springer US.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed–Solomon and algebraic-geometric codes. In Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280), pages 28–37. IEEE, 1998.

[Laz83] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In Computer algebra, volume 162 of LNCS, pages 146–156, Berlin, 1983. Springer. Proceedings Eurocal'83, London, 1983.

[Nie14] Johan Sebastian Rosenkilde Nielsen. Power decoding of reed-solomon codes revisited. In Raquel Pinto, Paula Rocha Malonek, and Paolo Vettori, editors, Coding Theory and Applications, 4th International Castle Meeting, ICMCTA 2014, Palmela Castle, Portugal, September 15-18, 2014, volume 3 of CIM Series in Mathematical Sciences, pages 297–305. Springer, 2014.

[Nie18] Johan Sebastian Rosenkilde Nielsen. Power decoding Reed-Solomon codes up to the Johnson radius. Advances in Mathematics of Communications, 12(1):81, 2018.

[Spa12] Pierre-Jean Spaenlenhauer. Résolution de systèmes multi-homogènes et determinantiels. PhD thesis, Univ. Pierre et Marie Curie- Paris 6, October 2012.

[SSB10] Georg Schmidt, Vladimir Sidorenko, and Martin Bossert. Syndrome decoding of Reed-Solomon codes beyond half the minimum distance based on shift-register synthesis. IEEE Trans. Inf. Theory, 56(10):5245–5252, 2010.

[Sud97] Madhu Sudan. Decoding of Reed–Solomon codes beyond the error–correction bound. J. Complexity, 13(1):180–193, 1997.