On Rational Interpolation-Based List-Decoding and List-Decoding Binary Goppa Codes

Beelen, Peter ; Høholdt, Tom; Nielsen, Johan Sebastian Rosenkilde; Wu, Yingquan

Published in:
I E E E Transactions on Information Theory

Link to article, DOI:
10.1109/TIT.2013.2243800

Publication date:
2013

Document Version
Peer reviewed version

Citation (APA):
Beelen, P., Høholdt, T., Nielsen, J. S. R., & Wu, Y. (2013). On Rational Interpolation-Based List-Decoding and List-Decoding Binary Goppa Codes. *I E E E Transactions on Information Theory*, *59*(6), 3269-3281. https://doi.org/10.1109/TIT.2013.2243800

On Rational-Interpolation Based List-Decoding and List-Decoding Binary Goppa Codes

Peter Beelen, Tom Høholdt, *Fellow, IEEE,* Johan S. R. Nielsen, and Yingquan Wu, *Senior Member, IEEE*

***Abstract*—We derive the Wu list-decoding algorithm for Generalised Reed-Solomon (GRS) codes by using Gröbner bases over modules and the Euclidean algorithm (EA) as the initial algorithm instead of the Berlekamp-Massey algorithm (BMA). We present a novel method for constructing the interpolation polynomial fast. We give a new application of the Wu list decoder by decoding irreducible binary Goppa codes up to the binary Johnson radius. Finally, we point out a connection between the governing equations of the Wu algorithm and the Guruswami-Sudan algorithm (GSA), immediately leading to equality in the decoding range and a duality in the choice of parameters needed for decoding, both in the case of GRS codes and in the case of Goppa codes.**

***Index Terms*—list decoding, rational interpolation, list size, Reed-Solomon code, Goppa code, Johnson radius**

I. Introduction

IN [1], Wu presented a decoding algorithm for Generalised Reed-Solomon (GRS) codes which decodes beyond half the minimum distance. Just like the Guruswami-Sudan algorithm (GSA) [2], the decoder might return a list of candidate codewords, justifying the term *list decoder*. The two algorithms share many other properties, most notably the decoding radius: they can both decode an $[n,\ k,\ n-k+1]$ GRS code up to $n-\sqrt{n(k-1)}$; the so-called Johnson radius.

The Wu list decoder reuses the output of the Berlekamp-Massey algorithm (BMA). The BMA has long been used for solving the Key Equation of GRS codes [3] whenever the number of errors is less than half the minimum distance. Wu noted that the result of the BMA still reveals crucial information about solutions to the Key Equation when more errors have occurred, and used this for setting up a rational interpolation problem. This problem can be solved by a generalisation of the core of the GSA, which solves a similar problem for polynomials.

The equivalence of the BMA and a special utilisation of the extended Euclidean algorithm (EA) is well-studied, e.g. [4]–[6]. Inspired especially by Fitzpatrick [4], we recast the Key Equation and the first part of the Wu list decoder into the language of Gröbner bases over certain modules, making it possible to use the EA; a generally more flexible and algebraic approach than the BMA.

The rational interpolation problem is attacked by first constructing an interpolation polynomial. This can be done by solving a large linear system of equations, but that is prohibitively slow. We give a fast method for constructing the interpolation polynomial which has the same asymptotic complexity as the fastest known methods for polynomial interpolation as used in the GSA. This also renders the Wu list decoder as fast as the fastest variants of the GSA.

The decoding radius and the choice of auxiliary parameters in the Wu list decoder is governed by having to satisfy a certain inequality, just as in the GSA; we point out that in the case of decoding GRS codes, the inequality in the Wu list decoder *becomes* the governing inequality by a simple change of variables, immediately implying that they have the same decoding radius and always use the same list size.

We show how the Wu list decoder can be adapted to decode binary Goppa codes. The algorithm is a continuation of the Patterson decoder [7], and the adaption of the Wu list decoder to this case is particularly simple due to the use of the EA instead of the BMA. Similarly to the case of GRS codes, we point out a connection between the governing inequality of the decoding parameters and the equation for the GSA with the Kötter-Vardy multiplicity assignment method (GSA+KV). This immediately yields that the methods have the same decoding radii, namely up to the binary Johnson radius $\frac{1}{2}n-\frac{1}{2}\sqrt{n(n-2d)}$, where n is the length and d the designed minimum distance of the Goppa code. Using our fast interpolation method, also this algorithm is as fast or faster than the previously known algorithms with the same decoding radius.

A. Related Work

The Wu list decoder is fairly recent and not much work has been done on it yet. In both Trifonov [8], [9] and Ali and Kuijper [10], an algorithm very closely related to the Wu list decoder for GRS codes is reached using a Gröbner basis description. The algorithm, however, revolves around two polynomials $G(x)$ and $R(x)$, where $G(x)$ is defined as the polynomial vanishing at the evaluation points of the code and $R(x)$ is the Langrange polynomial through the received word coordinates at the evaluation points. These polynomials are of higher degree than those used by the original Wu list decoder: the syndrome polynomial and a "modulus" x^{n-k}. More importantly, they are quite specific to the setting of decoding GRS codes.

We take a slightly different approach, closer to the original one by Wu. We essentially show how rational interpolation can help in solving Key Equations; that is, equations of the form

$$\gamma(x)q(x) \equiv \delta(x) \mod p(x)$$

where p, q are known polynomials, and one seeks γ and δ of low degree while additionally having certain knowledge on the evaluations of γ and δ. In the special case of GRS codes, this is exactly what the Wu list decoder does, but our description also immediately makes it clear that this can be used for binary Goppa codes.

The construction of the interpolation polynomial in the GSA is one of the most computationally expensive parts of that algorithm. A fast method for this is by Beelen and Brander [11] which refines one by Lee and O'Sullivan [12]; the main gain comes from solving the core polynomial-matrix problem using a faster method by Alekhnovich [13]. There is an even faster method for this matrix problem by Giorgi et al. [14], and using this in [11] yields the fastest known way of constructing the interpolation polynomial. Bernstein uses essentially the same approach for his GSA variant and achieves the same speed [15], both for Reed-Solomon codes and alternant codes; see also below. We show how this approach can be extended for rational interpolation, which ultimately leads to the Wu list decoder having the same asymptotic complexity as the GSA.

Binary Goppa codes have long been known to have much better minimum distances than their underlying GRS codes: if constructed with Goppa polynomial of degree t, the minimum distance is at least $2t + 1$, while it's GRS code has minimum distance $t + 1$, see e.g. [16]. Patterson's classic decoding algorithm utilises the binary property to decode t errors [7], but recent advances in list decoding allows decoding up to the binary Johnson radius $J_2 = \frac{1}{2}n - \frac{1}{2}\sqrt{n(n-4t-2)} > t$, where n is the length of the code.

Simply list decoding the underlying GRS code only reaches $n - \sqrt{n(n-t-1)} < t$, so this is not sufficient. However, by considering the Goppa code as one constructed with a degree $2t$ Goppa polynomial by utilising the identity of [17], and then using the GSA+KV, one reaches J_2, see e.g. [18] or [19, Section 9.6]. Alternatively, one can with the identity of [17] use Bernstein's decoder for alternant codes which works in a manner closely related to the GSA+KV [15].

The Kötter-Vardy method does not directly translate to the Wu list decoder, so a different approach is required. Our algorithm continues the original insights by Patterson by rewriting the Key Equation of the Goppa code into a reduced one of only half the degrees. This combined with list decoding turns out to also reach J_2.

B. Organisation

The remainder of this article is organised as follows: The introduction ends with some notation and notes on the modules that will be considered. In Section II we describe how solutions to certain Key Equation-like equations can be described using these modules, and how the EA can find these. In Section III, we introduce the problem of rational interpolation as well as a method to solve it for some parameters. We then show how the solution of the rational interpolation problem can be computed with low complexity. These two theoretical sections are then utilised in sections IV and V for decoding GRS codes and binary Goppa codes respectively. For each of those code families, we analyse the parameters needed for solving the associated rational interpolation problem, and we compare asymptotic running times with previous decoding methods.

C. Notation

Let $\mathbb{F}$ be a finite field. Define $R \subset \mathbb{F}[x, y]$ as all bivariate polynomials over $\mathbb{F}$ with y-degree at most 1. In this article, we will be considering $\mathbb{F}[x]$-modules that are subsets of R. Such a module could just as well be regarded as a subset of $\mathbb{F}[x] \times \mathbb{F}[x]$; however, using bivariate polynomials does give certain notational advantages.

We can define term orders as well as Gröbner bases over such modules. These definitions follow the general intuition from Gröbner bases over polynomial ideals. For an extensive presentation, see e.g. [20].

One thing to keep in mind is that term orders over $\mathbb{F}[x]$ sub-modules of R differs slightly from term orders over the polynomial ring $\mathbb{F}[x, y]$. For instance, the weighted degree term order giving x weight 1 and y weight 0, as well as lexicographically ordering x before y, is a valid module term order for these modules, while it is not valid over $\mathbb{F}[x, y]$.

For our discussions on modules and term orders, we define the following notational short-hands, where $<$ is a module term order and $h(x, y) \in R$:

- $[h_1, \dots, h_t] \triangleq \left\{ \sum_{i=1}^{t} a_i(x)h_i(x, y) \mid a_i(x) \in \mathbb{F}[x] \right\}$ is the $\mathbb{F}[x]$-module generated by $h_1, \dots, h_t \in \mathbb{F}[x, y]$.
- $\Delta f \triangleq \deg f(x)$ for $f(x) \in \mathbb{F}[x]$. Also define $\Delta f = -\infty$ when $f(x) = 0$.
- $\text{LT}_< h$ is the leading term of h wrt. $<$.
- $\Delta^x_<(h) \triangleq x\deg(\text{LT}_< h)$, where $x\deg(x^i y^j) = i$.
- $\Delta^y_<(h) \triangleq y\deg(\text{LT}_< h)$, where $y\deg(x^i y^j) = j$.

Note in particular here that the $\Delta^y_<(h)$ of an $h \in R$ is *not* the usual y-degree of h, but instead the y-degree of its leading term. In a sense, it describes the *position* of the leading term in h.

II. The Euclidean algorithm and Gröbner bases

Consider the following problem generalised from the Key Equation of algebraic coding theory: we are given two polynomials $p(x), q(x)$, and we seek two other polynomials $\gamma(x), \delta(x)$ of relatively low degrees which satisfy

$$\gamma(x)q(x) \equiv \delta(x) \mod p(x) \tag{1}$$

This equation alone might not be sufficient to uniquely determine $\gamma(x)$ and $\delta(x)$, but we would still like to gather as much information from the above equation as possible, in a certain sense.

Consider now the set $M = [p(x), y - q(x)] \in \mathbb{F}[x, y]$ as a module over $\mathbb{F}[x]$. We easily see that the polynomial $\delta(x) - y\gamma(x)$ is in M by using the above congruence:

$$\begin{aligned}\delta(x) - y\gamma(x) &= (\gamma(x)q(x) - w(x)p(x)) - y\gamma(x) \\ &= -\gamma(x)(y - q(x)) - w(x)p(x)\end{aligned}$$

for some polynomial $w(x)$. We might therefore study M in order to get a good description of $\gamma(x)$ and $\delta(x)$; we could, for example, seek a basis for M in which $\delta(x) - y\gamma(x)$ described in this basis has coefficients of low degree. As we will see, this can be given by a Gröbner basis under a certain module term order.

For a given ordering, we have the following easy condition for a generating set to be a Gröbner basis for the considered type of modules:

Proposition 1. *Let $M = [p(x), y - q(x)]$ be a module over $\mathbb{F}[x]$ for two polynomials $p(x), q(x)$ and let $<$ be a module term order. A set $G = \{h_1(x, y), h_2(x, y)\}$ is a Gröbner basis of M under $<$ if and only if $[G] = M$ and $\Delta^y_<(h_1) \neq \Delta^y_<(h_2)$.*

Proof: Follows straight-forwardly by applying Buchberger's S-criterion. □

For any $\mu \geq 0$, define now the module term order $<_\mu$ as the $(1, \mu)$ weighted-degree ordering of (x, y) with $x > y$. For example, $x^{\mu-1} <_\mu y <_\mu x^\mu$. We can now characterise the form of a Gröbner basis for M under this module term order, as well as the form of $\delta(x) - y\gamma(x)$ in this basis, given a limit on the degree of γ:

Proposition 2. *Let $G = \{h_1(x, y), h_2(x, y)\}$ be a Gröbner basis for $M = [p(x), y - q(x)]$ under $<_\mu$ with $\Delta^y_{<_\mu}(h_1) = 0$. Then $\Delta^x_{<_\mu}(h_1) + \Delta^x_{<_\mu}(h_2) = \Delta p$.*
Furthermore, if $\delta(x) - y\gamma(x) \in M$, then there exist polynomials $f_1(x), f_2(x)$ such that

$$\delta(x) - y\gamma(x) = f_1(x)h_1(x, y) + f_2(x)h_2(x, y)$$

If $\delta(x) <_\mu y\gamma(x)$ then these polynomials satisfy

$$\begin{aligned}\Delta f_1 &\leq \Delta\gamma + \mu - \Delta^x_{<_\mu}(h_1) - 1 \\ \Delta f_2 &= \Delta\gamma - \Delta^x_{<_\mu}(h_2)\end{aligned}$$

If $\delta(x) >_\mu y\gamma(x)$ then they instead satisfy

$$\begin{aligned}\Delta f_1 &= \Delta\delta - \Delta^x_{<_\mu}(h_1) \\ \Delta f_2 &\leq \Delta\delta - \mu - \Delta^x_{<_\mu}(h_2)\end{aligned}$$

Proof: Let us first prove the degree bounds on h_1 and h_2. Write $h_1(x, y) = h_{10}(x) + yh_{11}(x)$ and $h_2(x, y) = h_{20}(x) + yh_{21}(x)$. Note that $h_{11}(x)$ and $h_{21}(x)$ are coprime since some linear combination of them gives 1, as $y - q(x) \in M$. Then $f(x) = h_{21}(x)h_1(x, y) - h_{11}(x)h_2(x, y) \in M$ and does not contain y, and is the lowest degree polynomial in M to do so; this must be $cp(x)$ for some $c \in \mathbb{F}$, given the definition of M. Therefore $\Delta f = \Delta p$. However, by expanding the expression for f, we get

$$\begin{aligned}\Delta f &= \Delta\big(h_{21}(x)h_{10}(x) - h_{11}(x)h_{20}(x)\big) \\ &= \Delta(h_{21}(x)h_{10}(x))\end{aligned}$$

where we have used $\Delta^y_{<_\mu}h_1 = 0$ and $\Delta^y_{<_\mu}h_2 = 1$, the latter implied by Proposition 1.

Now for the statement on $\delta(x) - y\gamma(x)$. It is clear that f_1, f_2 satisfying the first of the equations exist, but we need to show the degree bounds. Assume first $\delta(x) <_\mu y\gamma(x)$. f_1, f_2 can be found by the division algorithm, so we consider how this would run. As $\delta(x) <_\mu y\gamma(x)$, we know that h_2 will be used as a divisor first, and it will divide so as to cancel the leading term; this first division therefore determines the degree of f_2 to be $\Delta\gamma - \Delta h_{21} = \Delta\gamma - \Delta^x_{<_\mu}h_2$. We might then perform more divisions by h_2 until at one point we use h_1; by then the remainder will be reduced to some $\grave{\delta}(x) - y\grave{\gamma}(x)$ with also $\grave{\delta}(x) >_\mu y\grave{\gamma}(x)$, and this division then determines the maximal degree of f_1 to $\Delta\grave{\delta} - \Delta h_{10}$. The division algorithm ensures us that the iterations has "decreased" the remainder, i.e. $\grave{\delta}(x) - y\grave{\gamma}(x) <_\mu \delta(x) - y\gamma(x)$ and therefore $\grave{\delta}(x) <_\mu y\gamma(x)$. As $<_\mu$ lexicographically orders x before y, we therefore must have $\Delta\grave{\delta}(x) \leq \Delta\gamma + \mu - 1$. In all, we get $\Delta f_1 \leq \Delta\gamma + \mu - \Delta^x_{<_\mu}h_1 - 1$. The case $\delta(x) >_\mu y\gamma(x)$ runs similarly. □

It turns out that the EA, if running on $p(x)$ and $q(x)$, in a certain manner produces Gröbner bases of the module M of module term order $<_\mu$. To prove this, we first need to remind of well-known results on the intermediate polynomials computed by the algorithm. For brevity, we don't present the EA algorithm in full, and consequently we can't prove the following lemma, but there are many good expositions on the algorithm which includes these results, e.g. Tilborg [21, Lemma 4.5.4] or Dornstetter [5].

Consider running the Extended Euclidean Algorithm (EA) on $p(x)$ and $q(x)$, and denote by $s_i(x)$ the remainder polynomial computed in each iteration i; that is, $s_0(x) = p(x)$, $s_1(x) = q(x)$ and $s_2(x), s_3(x), \ldots, s_N(x), s_{N+1}(x)$ will be the following remainders computed, where we know by the EA that $s_N(x) = \gcd(p, q)$ and $s_{N+1} = 0$. Then the EA in each iteration i also computes polynomials $u_i(x), v_i(x)$ such that $s_i(x) = u_i(x)p(x) + v_i(x)q(x)$. Furthermore, we have the following lemma, whose proof is easy by induction on the precise computations of the EA:

Lemma 3. *If the EA is run on polynomials $p(x), q(x)$ with $\Delta p > \Delta q$, the intermediate polynomials satisfy for each iteration $i = 1, \ldots, N + 1$:*

(i) *Δs_i is a decreasing function in i.*

(ii) $(-1)^i = u_i(x)v_{i-1}(x) - u_{i-1}(x)v_i(x)$

(iii) $s_i(x) = u_i(x)p(x) + v_i(x)q(x)$

(iv) $\Delta p = \Delta v_i + \Delta s_{i-1}$

We are now in a position to show how each iteration of the EA gives rise to a generating set for M:

Proposition 4. *Let the EA be run on two polynomials $p(x), q(x)$ with $\Delta p > \Delta q$. In each iteration i, let $G = \{h_1(x,y), h_2(x,y)\}$ with*

$$h_1(x,y) = s_{i-1}(x) - v_{i-1}(x)y$$
$$h_2(x,y) = s_i(x) - v_i(x)y$$

Then $[G] = M$ where $M = [p(x), y - q(x)]$.

Proof: Inserting the expression for $s_i(x)$ and s_{i-1} from Lemma 3 (iii), we get

$$\begin{pmatrix} h_1(x,y) \\ h_2(x,y) \end{pmatrix} = \begin{pmatrix} u_{i-1}(x) & -v_{i-1}(x) \\ u_i(x) & -v_i(x) \end{pmatrix} \begin{pmatrix} p(x) \\ y - q(x) \end{pmatrix}$$

Now $h_1(x,y), h_2(x,y)$ and $p(x), y-q(x)$ will be bases for the same module if and only if the determinant of the 2×2-matrix is a unit. But this is stated in Lemma 3 (ii). □

We can now wrap up and show the main result of this section:

Proposition 5. *Let $p(x), q(x)$ be two polynomials with $\Delta p > \Delta q$, and let $\mu \geq 0$ be an integer. If the EA is run on $p(x), q(x)$ and it is halted on the first iteration i where $\Delta s_i < \Delta v_i + \mu$, then $G = \{h_1(x,y), h_2(x,y)\}$ is a Gröbner basis of $M = [p(x), y - q(x)]$ with module term order $<_\mu$, where h_1, h_2 are chosen as in Proposition 4 for iteration i.*

Proof: Clearly there *is* a first iteration i where $\Delta s_i < \Delta v_i + \mu$, for $\Delta s_{N+1} = -\infty$ and $\Delta v_{N+1} \geq 0$. Thus, at least the $(N+1)$st iteration satisfies the requirement. Conversely, the 0'th iteration does not satisfy it as $\Delta s_0 = \Delta p$ and $\Delta v_0 = -\infty$. Now to show that G is a Gröbner basis. From Proposition 4 we know that $[G] = M$, so by Proposition 1 we only need to show that the leading terms of h_1 and h_2 have different y-degree under $<_\mu$. But by the choice of i, we have both $\Delta^y_{<_\mu}(h_1) = 0$ and $\Delta^y_{<_\mu}(h_2) = 1$. □

III. Rational Interpolation

We will now describe how to solve the problem of finding rational curves that go through at least some number of prescribed points. The method is a generalisation of the GSA [2], and first described by Wu [1]. The formulation of our main theorem, Theorem 6, avoids some special handling of points at infinity and is due to Trifonov [9].

We are basically interested in a rational expression $\frac{f_2(x)}{f_1(x)}$ with numerator and denominator of low degrees, which goes through at least some τ out of n points $\big((x_0, \beta_0), \ldots, (x_{n-1}, \beta_{n-1})\big)$ where all $x_i \in \mathbb{F}$ while $\beta_i \in \mathbb{F} \cup \{\infty\}$. To handle the points at infinity, we can instead consider these as partially projective points $(x_i, y_i : z_i)$ with $\frac{y_i}{z_i} = \beta_i$ whenever $\beta_i \neq \infty$ and $(y_i, z_i) = (1, 0)$ otherwise.

In this language, the interpolation amounts to finding low-degree polynomials $f_1(x)$ and $f_2(x)$ such that for at least τ values of i, we have $y_i f_1(x_i) - z_i f_2(x_i) = 0$. The following theorem is a paraphrasing of [9, Lemma 3]; we omit the proof which is a generalisation of the proof of [2, Lemma 4].

First a notational short-hand: For a $Q \in \mathbb{F}[x,y,z]$, we define

$$\Delta_{(w_x,w_y,w_z)} Q(x,y,z) \triangleq \max\{i w_x + j w_y + h w_z \\ \mid \alpha x^i y^j z^h \text{ is a monomial of } Q(x,y,z)\}$$

That is, $\Delta_{(w_x,w_y,w_z)} Q(x,y,z)$ is the (w_x, w_y, w_z)-weighted degree of Q. Now the theorem:

Theorem 6. *Let ℓ, s and τ be positive integers, and let $\{(x_0, y_0, z_0), \ldots, (x_{n-1}, y_{n-1}, z_{n-1})\}$ be n points in $\mathbb{F}^3$ where for all i either y_i or z_i is non-zero. Assume that $Q(x,y,z) = \sum_{i=0}^{\ell} Q_i(x) y^i z^{\ell-i}$ is a non-zero partially homogeneous trivariate polynomial such that (x_i, y_i, z_i) are zeroes of multiplicity s for all $i = 0, \ldots, n-1$, and $\Delta_{(1,w_2,w_1)} Q < s\tau$, for two $w_1, w_2 \in \mathbb{R}_+ \cup \{0\}$. For any two coprime polynomials $f_1(x), f_2(x)$ satisfying $\Delta f_1 \leq w_1$, $\Delta f_2 \leq w_2$, as well as $y_i f_1(x_i) + z_i f_2(x_i) = 0$ for at least τ values of i. Then $(y f_1(x) + z f_2(x)) \mid Q(x,y,z)$.*

As with the GSA, such a trivariate polynomial can be constructed by setting up and solving a system of linear equations. Each point to go through with multiplicity s corresponds to a similar requirement in a bivariate polynomial (see e.g. [9, Lemma 1]), and therefore gives rise to $\frac{1}{2}s(s+1)$ linear equations, so the total number of equations is given by $\frac{1}{2}ns(s+1)$. The number of coefficients of Q – and therefore variables of the equation system – is at least $\sum_{i=0}^{\ell} s\tau - i w_2 - (\ell - i) w_1$; it is exactly this whenever all the terms in the sum are non-negative, but it can actually be more when some of them are negative. Expanding and collecting, we therefore have that at least any $n, \tau, w_1, w_2, \ell, s$ which satisfy:

$$\tfrac{1}{2}ns(s+1) < s\tau(\ell+1) - \tfrac{1}{2}\ell(\ell+1)(w_1 + w_2) \qquad (2)$$

allow for a construction of a satisfactory Q.

It is easy to see that Q can have at most ℓ factors of the form given in the theorem, as its y-degree is ℓ. For this reason, particularly inspired by its use for decoding and in concordance with the GSA, it is called the *(designed) list size*.

We are mostly interested in knowing for which values of n, τ and w_1, w_2 we can select s and ℓ such that the above is satisfied. For rational interpolation in general, a minimal selection of s and ℓ given these parameters is done in [8], so we will not repeat it here. When we will later use rational interpolation in the application of decoding in sections IV and V, we will show a relation between the parameter choices of the particular instances of rational interpolation and similar instances of polynomial interpolation using the GSA respectively GSA+KV, and this turns out to immediately give us bounds on τ as well as values for s and ℓ.

Theorem 6 parallels a result for polynomial interpolation as used in the GSA, see e.g. [2, Lemma 5]. However, for the application of decoding, it is not quite enough; when we later need to solve a rational interpolation problem for decoding, we seek f_1 and f_2 which interpolate the error positions, and therefore an unknown number of points, but their maximal degrees increase with the number of points they interpolate. This means that we can't use Theorem 6 directly: setting τ low while the allowed degrees of f_1, f_2 high would not allow us to construct Q, while setting τ high would not guarantee that we found f_1 and f_2 when only few points were interpolated. Luckily, we have the following lemma which says that the Q we construct for high τ will also find f_1 and f_2 that interpolate fewer points, as long as their degrees decrease appropriately:

Lemma 7. *Let $Q(x,y,z)$ satisfy the requirements of Theorem 6 for some $(\tau, \ell, s, w_1, w_2)$. Then $Q(x,y,z)$ also satisfies the requirements for $(\tilde{\tau}, \ell, s, \tilde{w}_1, \tilde{w}_2)$ as long as*

$$\min\{w_1 - \tilde{w}_1 \ , \ w_2 - \tilde{w}_2\} \geq \tfrac{s}{\ell}(\tau - \tilde{\tau})$$

Proof: As the interpolation points and multiplicity as well as the list size have not changed, we only need to show $\Delta_{(1,\tilde{w}_2,\tilde{w}_1)}Q < s\tilde{\tau}$ We have:

$$\begin{aligned} \Delta_{(1,\tilde{w}_2,\tilde{w}_1)}Q &\leq \Delta_{(1,w_2,w_1)}Q \\ &\quad - \min\{i(w_2 - \tilde{w}_2) + (\ell - i)(w_1 - \tilde{w}_1) \mid 0 \leq i \leq \ell\} \\ &< s\tau - \ell \min\{w_1 - \tilde{w}_1, \ w_2 - \tilde{w}_2\} \end{aligned}$$

Therefore Q satisfies the degree constraints whenever

$$\begin{aligned} s\tau - \ell \min\{w_1 - \tilde{w}_1, \ w_2 - \tilde{w}_2\} \leq s\tilde{\tau} \qquad &\iff \\ \min\{w_1 - \tilde{w}_1, \ w_2 - \tilde{w}_2\} \geq \frac{s}{\ell}(\tau - \tilde{\tau}) \qquad & \end{aligned}$$ □

A. Fast interpolation

As mentioned, the interpolation polynomial $Q(x,y,z)$ can be constructed by setting up and solving a linear system of equations. However, without more thought, this would have a cubic running time in the size of the equation system, which is prohibitively slow. In this section, we describe a fast way to construct the polynomial, building heavily upon ideas from the similar problem in the GSA, in particular Lee and O'Sullivan [12] and the subsequent refinement in Beelen and Brander [11]

In the context of Theorem 6, consider given values of the parameters. We will assume that $\ell \geq s$; in later sections where we apply rational interpolation, this turns out always to be the case. Consider now the set $W \subset \mathbb{F}[x,y,z]$ consisting of *all* polynomials homogeneous of degree ℓ in y and z, and which interpolate the n points $\{(x_0,y_0,z_0), \dots, (x_{n-1},y_{n-1},z_{n-1})\}$, each with multiplicity at least s. Our goal is then to find a non-zero $Q \in W$ of lowest possible $(1,w_2,w_1)$-weighted degree. It is easy to see that W is an $\mathbb{F}[x]$-module. The approach is to give an explicit basis for W, represent this basis as a matrix over $\mathbb{F}[x]$ and then use an off-the-shelf algorithm for finding the "shortest" vector in that matrix, "short" being defined appropriately. This will correspond to a satisfactory interpolation polynomial.

Let us assume without loss of generality that each $z_i \in \{0,1\}$. Define the following polynomials which will turn out to play a crucial role: $R_y(x)$ and $R_z(x)$ will be the Lagrange polynomials interpolating (x_i,y_i) respectively (x_i,z_i), $i = 0,\dots,n-1$. Define also $G(x) = \prod_{i=0}^{n-1}(x - x_i)$ as well as $g_z(x) = \gcd(G, R_z)$. Now, there must exist $\lambda_1(x), \lambda_2(x) \in \mathbb{F}[x]$ such that $g_z(x) = \lambda_1(x)G(x) + \lambda_2(x)R_z(x)$. Define $\Upsilon(x) = \big(\lambda_2(x)R_y(x) \bmod G(x)\big)$, considered in $\mathbb{F}[x]$. Note that $\Upsilon(x_i) = \lambda_2(x_i)y_i$ for all $i = 0,\dots,n-1$. We begin with a small lemma:

Lemma 8. *Let $P(x,y,z) \in W$ and $P(x,y,z) = \sum_{j=0}^{\ell} P_j(x)y^j z^{\ell-j}$. Then $g_z(x)^{j-(\ell-s)} \mid P_j(x)$ for $j = \ell-s+1,\dots,\ell$.*

Proof: Let $L = \{x_i | z_i = 0\}$ so $g_z = \prod_{i \in L}(x - x_i)$. As P interpolates the points (x_i,y_i,z_i) with multiplicity s, $P(x+x_i, y+y_i, z+z_i)$ can have no monomials of total degree (in x,y and z) less than s. For $x_i \in L$ we have $P(x+x_i, y+y_i, z+z_i) = \sum_{j=0}^{\ell} P_j(x+x_i)(y+y_i)^j z^{\ell-j}$. All the terms in the sum have different z-degree, so nothing between these terms cancels, and so each can have no monomials of total degree less than s. In particular, since $z_i = 0$ we have $y_i \neq 0$, so multiplying out the power of $y + y_i$, this implies that $P_j(x+x_i)y_i^j z^{\ell-j}$ has no monomials of degree less than s. But then for $j = \ell-s+1,\dots,\ell$ we get $x^{j-(\ell-s)} \mid P_j(x+x_i)$. This implies the sought. □

The main result is the basis for W; it looks complicated, but the important thing is that it is directly calculable given the rational interpolation problem. We introduce for any $x \in \mathbb{R}$ the function $\mathrm{pos}(x) := \max(x,0)$. Note the easy identity $\mathrm{pos}(x) - \mathrm{pos}(-x) = x$. For the proof, we also use the phrase "leading monomial' of a trivariate polynomial $P(x,y,z)$ as the monomial of highest y-degree when P is regarded over $\mathbb{F}[x][y,z]$, and the "leading coefficient" is the $\mathbb{F}[x]$-coefficient of the leading monomial.

Theorem 9. *Let for $j = 0,\dots,\ell$*

$$\begin{aligned} B^{(j)} = (g_z y - \Upsilon z)^{\mathrm{pos}(s-j)}(yz - R_y z^2)^{j-\mathrm{pos}(j-(\ell-s))-\mathrm{pos}(j-s)} \\ (z\tfrac{G}{g_z})^{\mathrm{pos}(j-(\ell-s))} y^{\mathrm{pos}(\ell-s-j)} z^{\mathrm{pos}(j-s)} \end{aligned}$$

Then $W = \big[B^{(0)},\dots,B^{(\ell)}\big]$.

Proof: First, it should be proved that each $B^{(j)}$ are of total degree ℓ in y and z. By summing all the terms' exponents, counting each $yz - R_y z^2$ twice, and using the identity for $\mathrm{pos}(\cdot)$ given above, one sees this is so.

To show that each the $B^{(j)}$ are in W, note first that $yz - R_y z^2$ interpolate all (x_i,y_i,z_i). This is also true for $z\frac{G}{g_z}$ and $g_z y - \Upsilon z$, since either $z_i = 0$, whereby they obviously both evaluate to 0, or $x_i \notin L$ which gives $\frac{G(x)}{g_z(x)}|_{x=x_i} = 0$ as well as

$$\begin{aligned} g_z(x_i)y_i - \Upsilon(x_i)z_i &= (\lambda_1(x_i)G(x_i) + \lambda_2(x_i))y_i - \lambda_2(x_i)y_i \\ &= 0 \end{aligned}$$

For each $B^{(j)}$ to interpolate the points with multiplicity at least s, we need only to verify that the sum of the exponents of the three terms $g_z y - \Upsilon z$, $yz - R_y z^2$ and $z\frac{G}{g_z}$ is at least s for all generators; this is quickly seen to be true.

We need then only to show that any $P \in W$ can be expressed as an $\mathbb{F}[x]$-combination of the $B^{(j)}$. There are two cases to consider, $\ell - s \leq s$ and $\ell - s > s$. We will only show the latter case, and the former follows similarly. So assume $\ell - s > s$. Observe that $B^{(j)}$ has y-degree exactly $\ell - j$. The proof now basically follows the multivariate division algorithm on

P under lexicographical ordering $y > z > x$; i.e. dividing with the aim of lowering the y-degree.

First observe that the leading coefficient of $B^{(0)}$ is $g_z(x)^s$. By Lemma 8, we can perform polynomial division of P by $B^{(0)}$ and get a remainder $P^{(1)}(x, y, z)$ of y-degree at most $\ell - 1$. As $B^{(0)} \in W$ so is $P^{(1)} \in W$. We can continue as such with $B^{(j)}$ for $j = 1, 2, \ldots, s-1$, as each of these $B^{(j)}$ has leading coefficient $g_z(x)^{s-j}$ and Lemma 8 promises that the remainders will keep having leading term divisible by exactly this. We thus end with a remainder $P^{(s)}$ with y-degree at most $\ell - s$ and in W.

As $\ell - s > s$ then for $j = s, \ldots, \ell - s$ we have $B^{(j)}(x, y, z) = (yz - R_y z^2)^s y^{\ell-s-j} z^{j-s}$. They all have leading coefficient 1, so we can reduce $P^{(s)}$ with $B^{(s)}$, reduce the remainder of that with $B^{(s+1)}$ and so forth, until we arrive at a remainder $P^{(\ell-s+1)}$ with y-degree at most $s - 1$.

Still we have $P^{(\ell-s+1)} \in W$ so the (x_i, y_i, z_i) are all zeroes with multiplicity s. Therefore $P^{(\ell-s+1)}(x + x_i, y + y_i, z + z_i)$ has no monomials of degree less than s. Let $\overline{L} = \{x_i | z_i \neq 0\}$ and let $P^{(\ell-s+1)}(x, y, z) = \sum_{j=0}^{s-1} P_j^{(\ell-s+1)}(x) y^j z^{\ell-j}$. For $x_i \in \overline{L}$, we see by expanding the powers of both $y + y_i$ and $z + z_i$ that $P^{(\ell-s+1)}(x + x_i, y + y_i, z + z_i)$ has a monomial $P_{s-1}^{(\ell-s+1)}(x + x_i) y^{s-1} z_i^{\ell-s}$ which does not cancel with any other term. Therefore, $x \mid P_{s-1}^{(\ell-s+1)}(x + x_i) \iff (x - x_i) \mid P_{s-1}^{(\ell-s+1)}$. Collecting for all $x_i \in \overline{L}$, we get $\frac{G}{g_z} \mid P_{s-1}^{(\ell-s+1)}$. Note that, as $\ell - s > s$, then $B^{(\ell-s+1)}(x)$ has leading coefficient $\frac{G}{g_z}$. Thus, we can divide $P^{(\ell-s+1)}(x, y, z)$ by $B^{(\ell-s+1)}(x)$ and get remainder $P^{(\ell-s+2)}$ of y-degree at most $s - 2$.

Now, the exact same argument as above can be repeated for $P^{(\ell-s+2)}$, but one finds that $(x - x_i)^2$ must divide the leading coefficient for each $x_i \in \overline{L}$. Therefore, we can divide by $B^{(\ell-s+2)}$ whose leading coefficient is $(\frac{G}{g_z})^2$. We can continue this way with all the remaining $B^{(j)}$, until we find that the last remainder $P^{(\ell)}$ must be divisible by $(\frac{G}{g_z})^s z^\ell = B^{(\ell)}$. □

With a concrete basis for W in hand, we wish to find an element in W with lowest possible $(1, w_2, w_1)$-weighted degree. Write the $B^{(j)}$ of Theorem 9 as $B^{(j)}(x, y, z) = \sum_{i=0}^{\ell} B_i^{(j)}(x) y^i z^{\ell-i}$. Construct now the matrix $\Pi \in \mathbb{F}[x]^{(\ell+1)\times(\ell+1)}$ where the (j, i)'th entry is $B_i^{(j)}(x)$. The $B^{(j)}(x, y, z)$ thus constitute the rows of Π. In this manner, we can represent any basis of W as an $(\ell + 1) \times (\ell + 1)$ matrix, and any $P \in W$ can be represented as a vector in the row span of such a basis matrix.

Consider a vector V in the row-span of Π, and denote by $|V| := \max_{V_j \neq 0}\{\Delta V_j + j w_2 + (\ell - j) w_1\}$ where V_j is the j'th component of V. A shortest vector in Π under this metric will correspond to a polynomial in W which has the lowest possible $(1, w_2, w_1)$-degree. Any algorithm which can compute a shortest vector in the row-span of an $\mathbb{F}[x]$-matrix under this metric will therefore be usable to solve our problem.

The usual approach of such algorithms is to compute a so-called *row reduced* basis matrix, where the sum of the basis elements' lengths is minimal. It is well known that the shortest vector in the row space will be present in this reduced matrix, see e.g. [13], [22]. This problem is widely studied and it has several different guises and names: Gröbner basis reductions over free $\mathbb{F}[x]$-modules [12], row reduction of $\mathbb{F}[x]$-matrices [14], and basis reduction of $\mathbb{F}[x]$-lattices [23].

The fastest method in the literature for our purposes is due to Giorgi et al. in [14]. If θ is the highest degree of any polynomial in the initial basis matrix, and the basis matrix is $\nu \times \nu$, then the algorithm has complexity $O(\nu^\omega \theta \log^{O(1)}(\nu\theta))$, where $O(\nu^\omega)$ is the complexity for multiplying two $\nu \times \nu$ matrices with elements in $\mathbb{F}$. Trivially $\omega \leq 3$ but methods exist with $\omega < 2.4$ [24]. To bound the running time of applying the algorithm on our problem, we have the following:

Lemma 10. *In the context of Theorem 9 and the discussion above, the entries of* Π *all have degree at most* sn*.*

Proof: The entries of Π are all of the form $\beta g_z^{j_1} \Upsilon^{j_2} R_y^{j_3} (\frac{G}{g_z})^{j_4}$ where $\beta \in \mathbb{F}$ and j_1, j_2, j_3, j_4 are non-negative integers summing to at most s. The lemma follows as the four base polynomials are each of degree at most n. □

The algorithm in [14] does not directly support the different "column weights" that our vector metric demands, but this can be amended by first multiplying the j'th column of Π with $x^{j w_2 + (\ell - j) w_1}$ and then finding the usual row reduced basis. The powers of x can then be divided out from the resulting reduced basis afterwards. This does not change the complexity of the algorithm whenever $w_1, w_2 \in O(n)$, which follows if we assume $\tau^2 > n(w_1 + w_2)$; an assumption which turns out to be true for our applications in later sections. One should also note that for finite fields $\mathbb{F}$, the algorithm might need to calculate over an extension field, though without affecting the asymptotic running time, as pointed out by Bernstein [15]. This entire discussion can be distilled into the following:

Lemma 11. *For given values of the parameters of Theorem 6 where* $\ell \geq s$ *and* $\tau^2 > n(w_1 + w_2)$*, an algorithm exists to find a satisfactory interpolation polynomial in complexity* $O(\ell^\omega sn \log^{O(1)}(\ell n))$*.*

Proof: As soon as one has constructed Π, the result follows from Lemma 10 and the complexity of the algorithm in [14], so we just need to show that we can compute Π in the given speed. Let $M(\theta)$ be the complexity of multiplying two polynomials of degree θ. Computing R_y, R_z and G by Lagrangian interpolation can be done in complexity $O(M(n) \log n)$, see e.g. [25, p. 235]. Υ and g_z can be computed using the Euclidean algorithm in $O(n \log^2 n)$. For a polynomial of degree n, computing all the first s different powers of it can be done iteratively in $O(sM(sn))$. Each entry in Π is a multiple of $g_z, R_y, \frac{G}{g_z}$ and Υ to a combined power of s, so after each of their s powers have been computed, each of the $O(\ell^2)$ entries in Π can be computed in $O(M(sn))$. Using Schönhage-Strassen, we can set $M(\theta) = O(\theta \log \theta \log \log \theta)$, see e.g. [25, Theorem 8.23], and inserting this into the above, we see that Π can be computed in $O(\ell^2 M(sn)) \subset O(\ell^2 sn \log^{O(1)}(\ell n))$. □

Remark: Another algorithm that can be used to handle the interpolation problem is the row-reduction method of Alekhnovich [13], which also has been used in the interpolation method by Beelen and Brander. [11]. This method could also be used here but would yield the slightly worse running time $O(\ell^4 s \log^{2+o(1)}(\ell n))$. ■

After having computed the interpolation polynomial $Q(x, y, z)$, one needs to find factors of the form $yf_1(x) + zf_2(x)$ with $f_1, f_2 \in \mathbb{F}[x]$. Any such factor except z will also occur as an $\mathbb{F}(x)$ factor in the dehomogenised version of Q. Thus, any fast algorithm for computing this will suffice. In [1], Wu describes an extension to the root-finding method of Roth and Ruckenstein (RRR) [26] for finding $\mathbb{F}(x)$ roots of a $\mathbb{F}[x][y]$ polynomial: he remarks that simply applying the original RRR will find the truncated power series of each $\mathbb{F}(x)$ root; retrieving a long enough such series and applying a Padé approximation method like the BMA or the EA will retrieve the polynomial fraction. A divide-and-conquer speed-up of the RRR described by Alekhnovich in [13, Appendix] applies just as well to this extension[1]. We arrive at the following

Lemma 12. *In the context of Theorem 6, there exists an algorithm which finds all factors of $Q(x, y, z)$ of the form $yf_1(x) + zf_2(x)$ in complexity $O\big(\ell^2 sn \log(\ell n)^{2+o(1)}\big)$, where q is the cardinality of $\mathbb{F}$ and assuming $q \in O(n)$.*

Proof: The root-finding algorithm described in [1] will have the complexity of running the RRR followed by at most ℓ applications of the EA, each on a truncated power series of degree $O(\tau) \in O(n)$. The EA applications will have complexity $O(\ell n \log n)$ which is in the complexity of the lemma.

For running the RRR, Alekhnovich reports a complexity of $O(\ell^{O(1)}\theta \log \theta)$, where θ is the x-degree of $Q(x, y, z)$; however, his analysis can be improved: in the context of his proof, choose a fast factoring method over $\mathbb{F}[y]$, e.g. from [25, Theorem 14.14], and so set $f(1, \ell) = O(\ell M(\ell) \log(q\ell))$. The non-recursive cost of $f(\theta, \ell)$, i.e. the term $\ell^{O(1)}\theta$, can be improved to $\ell^2 M(\theta)$, as an upper bound cost of the ℓ different calculations of the shifts $Q(x, y_i + x^{d_i}\hat{y})$. Now the recursive bound has the improved solution $f(\theta, \ell) \in O(\ell^2 M(\theta) \log \theta + \theta \ell M(\ell) \log(q\ell))$. We have $\theta \in O(sn)$ and assume $q \in O(n)$ and thus arrive at the complexity of the lemma. □

An alternative factorisation method with roughly the same complexity is proposed by Bernstein in [15] by accommodating a more classical root finding method in $\mathbb{Z}[x]$ by Zassenhaus; see also [25, Chapter 15].

IV. Wu list decoding for Reed-Solomon codes

We can now derive the Wu list decoder in a succinct manner using the Euclidean algorithm instead of the Berlekamp-Massey algorithm (BMA). This derivation is inspired by Trifonov's derivation [9], though ours is slightly more general and uses shorter polynomials in the computations.

A. The codes

An $[n, k, d]$ Generalised Reed-Solomon (GRS) code over a finite field $\mathbb{F}_q$ is the set

$$\big\{\big(v_0\eta(\alpha_0), \dots, v_{n-1}\eta(\alpha_{n-1})\big) \mid \eta \in \mathbb{F}_q[x] \wedge \Delta\eta < k\big\}$$

for some n distinct non-zero $\alpha_0, \dots, \alpha_{n-1} \in \mathbb{F}_q$ as well as n non-zero $v_0, \dots, v_{n-1} \in \mathbb{F}_q$. The α_i are called *evaluation points* and the v_i *column multipliers*. It is easy to show that $d = n - k + 1$ and the code is therefore MDS. See e.g. [19] for a comprehensive introduction to GRS codes.

Consider a sent codeword $c = (c_0, \dots, c_{n-1})$ and a corresponding received word $r = (r_0, \dots, r_{n-1})$. Then the syndrome polynomial is computable by the receiver and can be defined as

$$S(x) = \sum_{i=0}^{n-k-1} x^i \sum_{j=0}^{n-1} r_j w_j \alpha_j^{d-2-i} \tag{3}$$

where $w_j = (v_j \prod_{h \neq j}(\alpha_j - \alpha_h))^{-1}$. If we denote the set of error locations by E, that is, $E = \{i \mid c_i \neq r_i\}$, we can define the error-locator and error-evaluator polynomials respectively, as follows[2]:

$$\Lambda(x) = \prod_{i \in E}(x - \alpha_i)$$
$$\Omega(x) = -\sum_{i \in E}(r_i - c_i)\alpha_i^{d-1} w_i \prod_{j \in E \setminus \{i\}} (x - \alpha_j)$$

Clearly, the receiver can quickly retrieve c from r if he constructs Λ and Ω, as the error locations are the roots of Λ, and the error values are the evaluations of Ω in the respective error location (up to a calculable scalar). Note that therefore $\gcd(\Lambda, \Omega) = 1$ as the elements of E are all the zeroes of Λ but definitely not zeroes of Ω. The three defined polynomials are related by the famous Key Equation (see e.g. [19] or [27]):

$$\Lambda(x)S(x) \equiv \Omega(x) \mod x^{d-1} \tag{4}$$

Many decoding algorithms solve this equation for Λ and Ω, and construct c from r using these. That is also what our list decoder will do.

[1] We are grateful to the anonymous reviewer for pointing out the extension of Alekhnovich to us as well as the improvement to its running time analysis.

[2] The reader familiar with the three polynomials might notice our slightly unorthodox definition of them; many sources use an error-locator which reveals the *inverse* error positions, i.e. $\Lambda(\alpha_i^{-1}) = 0$ iff the i'th position is in error. This also yields a slightly simpler syndrome polynomial. However, in the case of Goppa codes, the above definition of the error locator is more natural, and we have opted for consistency in this article by also using that here.

B. The list-decoding algorithm

Using the Key Equation and the results of Section III, we can construct a list decoder. By (4) as well as (1) on page 2 and the paragraphs following it, we know that $\Omega(x) - y\Lambda(x) \in M = [x^{d-1}, y - S(x)]$. If we run the EA on x^{d-1} and S, by Proposition 5, we get a Gröbner basis $G = \{h_1, h_2\}$ of M of module term order $<_\mu$ for any integer $\mu \geq 0$. We choose $\mu = 0$.

Let $\epsilon = |E|$ be the number of errors, unknown to the receiver. Then $\Delta\Omega < \Delta\Lambda = \epsilon$. As $\Delta\Lambda > \Delta\Omega$, then $y\Lambda(x) >_0 \Omega(x)$. Assume now that $\Delta^y_{<_0} h_2 = 1$ (and therefore $\Delta^y_{<_0} h_1 = 0$). Therefore, by Proposition 2, we know there exist polynomials $f_1, f_2 \in \mathbb{F}[x]$ such that

$$\begin{aligned} \Omega(x) - y\Lambda(x) &= f_1(x)h_1(x,y) + f_2(x)h_2(x,y) \\ \Delta f_1 &\leq \epsilon - d + \Delta^x_{<_0}(h_2) \\ \Delta f_2 &= \epsilon - \Delta^x_{<_0}(h_2) \end{aligned} \tag{5}$$

We see that whenever $\epsilon \leq \lfloor \frac{n-k}{2} \rfloor$, either the degree bound for f_1 or that for f_2 will be negative, and that one will then be zero. Therefore $\Omega(x) - y\Lambda(x)$ will be a multiple of either h_1 or h_2. As $\Delta^y_{<_0}(\Omega(x) - y\Lambda(x)) = 1$, it must be a multiple of h_2. However, as Λ and Ω are coprime, that multiple must be the constant that normalises h_2 to have leading coefficient 1, just as $\Lambda(x)$. This corresponds to the Sugiyama decoding algorithm [28].

In case neither h_1 nor h_2 is valid as $\Omega(x) - y\Lambda(x)$, we know that f_1 and f_2 are non-zero, so there are more errors than half the minimum distance; then we proceed exactly like regular Wu list decoding using BMA. We know that for at least ϵ values of $x_0 \in \{\alpha_0, \ldots, \alpha_{n-1}\}$, we have $\Lambda(x_0) = 0$, namely the error locations. Therefore, by (5), for at least those ϵ values of x_0, we have $f_1(x_0)h_{11}(x_0) + f_2(x_0)h_{21}(x_0) = 0$. Thus, for this to be a rational interpolation problem as in Section III, we just need to ascertain two properties: 1) that $h_{11}(x)$ and $h_{21}(x)$ never simultaneously evaluate to zero since they are coprime, as a linear combination of h_1 and h_2 equals $y - S(x) \in M$. 2) that f_1 and f_2 are coprime since Λ and Ω are.

From the results developed in Section III, we can therefore solve this rational interpolation problem for certain values of ℓ as well as the parameters n and d: we construct a partially homogeneous interpolation polynomial $Q(x, y, z)$ which has zero at all the points $(\alpha_i, h_{11}(\alpha_i), h_{21}(\alpha_i))$ for $i = 0, \ldots, n-1$. Under certain constraints on the degrees of $Q(x, y, z)$, then $yf_1(x) + zf_2(x)$ will be a factor of $Q(x, y, z)$. The following subsection looks closer at the possible choice of parameters to derive the upper bound on τ. The complete list decoder is listed in Algorithm 1.

Remark: There is a duality between the GSA and the Wu list decoder: in list decoding GRS codes with the GSA, one sets up an interpolation problem where the sought solution – the information word – will pass through those of the prescribed points that correspond to the error-free positions. Oppositely, here we seek f_1, f_2 that pass through those of the prescribed points that correspond to the errors positions. ■

C. Analysis of the parameters

It is clear that in Theorem 6, we should set w_1, w_2 equal to the bounds on $\Delta f_1, \Delta f_2$ in (5) for the case $\varepsilon = \tau$; so $w = w_1 + w_2 = 2\tau - d$. Note therefore that in this instance, w is always an integer. The main question is then for which τ we can select ℓ and s such that (2) is satisfied. Inserting the value for w and rearranging, (2) becomes

$$\frac{\tau}{n} < \frac{1}{(\ell+1)(\ell-s)} \left(\binom{\ell+1}{2} \frac{d}{n} - \binom{s+1}{2} \right) \tag{6}$$

Replacing s by $\ell - s$ this is exactly the equation governing the choice of parameters s, ℓ and τ in the GSA for the same values of n and d, see e.g. [19, Lemma 9.5]. This means that for all parameters of the GSA where the multiplicity is less than ℓ, this substitution applies, giving valid parameters for Algorithm 1.[3] We arrive at the following two lemmas:

Lemma 13. *Algorithm 1 can list decode for any $\tau < n - \sqrt{n(n-d)}$.*

Proof: For any given n, k, τ with τ less than the given bound, there exists a valid list size ℓ and multiplicity s_G such that the equation of the GSA is satisfied, and furthermore $s_G \leq \ell$, see e.g. [19, Lemma 9.5]. Except in the case $s_G = \ell$, the above duality applies and we are done, so assume $s_G = \ell$. As the governing equation of the GSA is satisfied, this means $\frac{\tau}{n} < \frac{1}{(\ell+1)\ell}\binom{\ell+1}{2}\frac{d}{n}$ so $\tau < \frac{d}{2}$, but in this case we are within minimum-distance decoding. Thus, Algorithm 1 will succeed in Step 3.

Thus we have established that for any given τ less than the given bound, we can select values of s and ℓ such that the sought f_1, f_2 can be found using Theorem 6 whenever $\epsilon = \tau$. Now, to be guaranteed to find them also whenever $\epsilon < \tau$, we also need to employ Lemma 7. This can be used if it is satisfied that

$$\min\{w_1 - \Delta f_1, w_2 - \Delta f_2\} \geq \tfrac{s}{\ell}(\tau - \epsilon)$$

Note that $w_1 - \Delta f_1 \geq \tau - \epsilon$ using (5). The same holds for $w_2 - \Delta f_2$. Therefore, the above is true at least if we satisfy

$$\tau - \epsilon \geq \tfrac{s}{\ell}(\tau - \epsilon) \qquad \Longleftrightarrow \qquad s \leq \ell$$

Thus, Lemma 7 guarantees that as long as $s \leq \ell$, then the $Q(x, y, z)$ we would construct satisfying the requirements of Theorem 6 will contain $yf_1(x) + zf_2(x)$ as a factor whenever $\epsilon \leq \tau$. But $s \leq \ell$ is satisfied as $s_G < \ell$ in all considered cases of the GSA and $s = \ell - s_G$ by the duality. □

Remark: This decoding radius – the so-called Johnson bound – is not the best one can achieve for a given GRS code: using the GSA+KV one can decode slightly further, namely up to the q-ary Johnson bound $\frac{q-1}{q}\big(n - \sqrt{n(n - \frac{q}{q-1}d)}\big)$, see e.g. [18] or [19, Section 9.6]. ■

Lemma 14. *For given n, k and τ with $\tau \geq \frac{n-k+1}{2}$, then ℓ and s are valid choices for the parameters for Algorithm 1 if*

[3] We are grateful to the anonymous reviewer for pointing out this relation to us.

and only if ℓ and $s_G = \ell - s$ are valid choices for the GSA. Furthermore, for any given ℓ, let s be the lowest possible choice of multiplicity for Algorithm 1 and s_G the lowest possible choice of multiplicity for the GSA; if $\tau < n/2$ then $s \leq s_G$, otherwise, $s \geq s_G$.

Proof: Only the last claim does not directly follow from the duality in parameter choice. Consider (6) governing the possible choice of s for Algorithm 1: rearranging to a second-degree equation in s and solving, we get that s/ℓ must be chosen from the interval $[T - \sqrt{D};\ T + \sqrt{D}]$, where $T = \frac{\tau}{n} + \frac{\tau - n/2}{n\ell}$ and D a discriminant whose precise expression is not important for us. Due to the duality between Algorithm 1 and the GSA, the corresponding interval for valid s_G/ℓ for the GSA will be $[1 - T - \sqrt{D};\ 1 - T + \sqrt{D}]$. In addition to residing in these respective intervals, we only require of s/ℓ and s_G/ℓ that s and s_G are positive integers less than ℓ. Therefore, whenever $\tau < n/2$ we have $T > \frac{1}{2}$, so the lowest possible choice of s in the former interval must be at most the lowest possible in the latter interval; oppositely for the case $\tau \geq n/2$. □

To concretely choose ℓ and s given n, k and τ, we can—due to the above lemma—use closed expressions designed for the GSA; e.g. [29, Eqs.(43-45)]. Alternatively, Trifonov and Lee give a simple analysis and expressions directly for the Wu list decoder in [8].

Algorithm 1 Wu list decoding GRS codes

```
Input: A GRS code $\mathcal{C}$ over $\mathbb{F}_q$ with parameters $n, k, d = n - k + 1$ and evaluation points $\alpha_0, \ldots, \alpha_{n-1}$, decoding radius $\tau < n - \sqrt{n(n-d)}$, and received word $r \in \mathbb{F}_q^n$.
Output: A list of all codewords in $\mathcal{C}$ within radius $\tau$ of $r$ or Fail if there are no such words.

1: Calculate the syndrome $S(x)$ from $r$ according to (3).
2: Run the EA on $x^{d-1}, S(x)$ and halt when $\Delta s_{\underline{i}} < \Delta v_i$, reusing the notation of Section II. Define $\tilde{h}_1(x) = -v_{\underline{i}-1}(x)$ and $\tilde{h}_2(x) = -v_i(x)$.
3: If $\tilde{h}_2$ is a valid error-locator of degree at most $d - \tau$, use it to correct $r$, and if this yields a word in $\mathcal{C}$, return this one word.
4: Otherwise, we seek $f_1, f_2$ according to (5). Set $w_1, w_2$ to the degree bounds of $f_1$ and $f_2$ for the case $\epsilon = \tau$, and calculate $\ell$ and $s$ to satisfy (6). Construct a $Q(x, y, z)$ satisfying the requirements of Theorem 6 using the points $\{(\alpha_i, \tilde{h}_1(\alpha_i), \tilde{h}_2(\alpha_i))\}_{i=0}^{n-1}$.
5: Find all factors of $Q(x, y, z)$ of the form $y f_1^\star(x) + z f_2^\star(x)$ where $f_1^\star$ and $f_2^\star$ have degree less than $w_1$ and $w_2$ respectively. Return Fail if no such factors exist.
6: For each such factor, construct $\Lambda^\star(x) = f_1^\star(x)\tilde{h}_1(x) + f_2^\star(x)\tilde{h}_2(x)$. If it is a valid error-locator, use it for correcting $r$. Return Fail if none of the factors yield error-locators
7: Return those of the corrected words that are in $\mathcal{C}$. Return Fail if there are no such words.
```

D. Complexity analysis

The complexity of the totality of Algorithm 1 is easily found using the results of Section III-A; note that $\tau^2 > nw$ whenever $\tau < n - \sqrt{n-d}$ so we can use Lemma 11. For simplicity, we will assume that $q \in O(n)$ where q is the cardinality of $\mathbb{F}$. In that case, as $\ell \geq s$, steps 4 and 5 can be computed in $O(\ell^{\omega+1} n \log^{O(1)}(\ell n))$. The remaining steps are of lower order: calculating $S(x)$ in step 1 can be done in $O(n \log n)$ using fast Fourier methods, and the EA in step 2 has complexity $O(n \log^2 n)$. Checking whether a polynomial is a valid error-locator takes at most $O(q)$, and in step 3 we check 2 such, while in step 6 we check at most ℓ such. Thus we have the following

Lemma 15. *If $q \in O(n)$ then Algorithm 1 has complexity $O(\ell^\omega s n \log^{O(1)}(\ell n))$.*

Using Lemma 14 we can compare running times with those for variants of the GSA. In this light, the above estimate is fast as the fastest GRS list-decoders based on the GSA. The bottle-neck is – as it is here – the construction of an interpolation polynomial. Beelen and Brander gave in [11] an algorithm for computing the interpolation polynomial in the GSA with complexity $O(\ell^5 n \log^2 n \log\log n)$, using an approach very close to the one here, and using a row reduction algorithm on an appropriate polynomial matrix. However, had they used the one by Giorgi et al. [14] instead of the slightly slower by Alekhnovich [13], they would have reached the same complexity as in Lemma 15, but using the value of s needed for the GSA.

It would therefore seem that, when the multiplicity for Algorithm 1 is smaller than the multiplicity for the GSA, Algorithm 1 would be faster than the GSA, though as we have only presented asymptotic analysis, one would need implementations to properly verify this. From Lemma 14 and its proof, we know that the multiplicity for Algorithm 1 is smallest whenever $\tau < n/2$ and that the difference to the multiplicity of the GSA increases with $\frac{\tau}{n}$.

Bernstein also gives a decoding algorithm in [15] with the same complexity, but his is a variant of the GSA+KV, and it can thus decode a GRS code to the slightly higher q-ary Johnson radius: $\frac{q-1}{q}\left(n - \sqrt{n(n - \frac{q}{q-1}d)}\right)$; see also Section V-D.

V. Wu list decoding binary Goppa codes

A. The codes

Consider an irreducible polynomial $g(x) \in \mathbb{F}_{2^m}[x]$ as well as n distinct elements of $\mathbb{F}_{2^m}$, $L = (\alpha_0, \ldots, \alpha_{n-1})$. Then the irreducible binary Goppa code $\Gamma(g, L)$ with Goppa polynomial g over L is the set

$$\left\{ (c_1, \ldots, c_n) \in \mathbb{F}_2^n \;\middle|\; \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \mod g(x) \right\}$$

This code has parameters $[n,\ \geq n - m\Delta g,\ \geq 2\Delta g + 1]$. A binary Goppa code $\Gamma(g, L)$ is a subfield subcode of an $[n,\ n - \Delta g,\ \Delta g + 1]$ GRS code over $\mathbb{F}_{2^m}$. It is also an alternant code. See e.g. [16] for a more complete description.

Consider a sent codeword $c = (c_0, \ldots, c_{n-1})$ and a corresponding received word $r = (r_0, \ldots, r_{n-1})$. For these codes, a natural definition of a syndrome polynomial is then

$$S(x) = \left(\sum_{i=0}^{n-1} \frac{r_i}{x - \alpha_i} \mod g(x) \right) \tag{7}$$

Like in the preceding section, we also define E, the error-locator and error-evaluator, the last being slightly simpler due to the binary field:

$$\begin{aligned} E &= \{ i \mid c_i \neq r_i \} \\ \Lambda(x) &= \prod_{i \in E} (x - \alpha_i) \\ \Omega(x) &= \sum_{i \in E} \prod_{j \in E \setminus \{i\}} (x - \alpha_j) \end{aligned}$$

Introduce also $\epsilon = |E|$ as the number of errors. We have again that $\gcd(\Lambda, \Omega) = 1$. It also turns out that the introduced polynomials satisfy a Key Equation [7]:

$$\Lambda(x) S(x) \equiv \Omega(x) \mod g(x) \tag{8}$$

Note that for a binary code, the receiver can decode immediately upon having calculated the error locator, even without the error evaluator; the error value is always 1.

B. The list-decoding algorithm

Now we could proceed exactly as in Section IV-B, and we would arrive at a list decoder correcting up to $n - \sqrt{n(n - \Delta g - 1)}$ errors. This is the same decoding radius reached by simply decoding the enveloping GRS code with the GSA or the Wu decoder. However, this radius is much less than Δg which is promised by the minimum distance of the binary Goppa code, and which can be corrected by Patterson's decoder [7].

Therefore, we proceed to rewrite the Key Equation in the same way as Patterson. In the following, it will be useful to assume $\epsilon < 2\Delta g$ as an initial and reasonable bound on our list decoder. Then, collecting even and odd terms, we can introduce polynomials $a(x), b(x)$ such that $\Lambda(x) = a^2(x) + x b^2(x)$ and satisfying $\Delta a \leq \lfloor \frac{\epsilon}{2} \rfloor$ and $\Delta b \leq \lfloor \frac{\epsilon - 1}{2} \rfloor$. Now, note from the definition of the polynomials that $\Omega(x)$ equals the formal derivative of $\Lambda(x)$, so we get $\Omega(x) = b^2(x)$ in this field of characteristic 2. The Key Equation thus becomes

$$\begin{aligned} (a^2(x) + x b^2(x)) S(x) \equiv b^2(x) \mod g(x) \iff \\ b^2(x)(x + S^{-1}(x)) \equiv a^2(x) \mod g(x) \end{aligned} \tag{9}$$

Note here that calculating the inverse of $S(x)$ modulo $g(x)$ is possible since $\Delta S < \Delta g$ and $g(x)$ is irreducible.

It might now be that $S^{-1}(x) \equiv x \mod g(x)$ in which case $a^2(x) \equiv 0 \mod g(x)$. As $g(x)$ is irreducible, $a(x)$ must be a multiple of $g(x)$, which means that $a(x) = 0$ as $\epsilon < 2\Delta g$. This implies $\Lambda(x) = x b^2(x)$, which is only a legal error locator if $0 \in L$ and $b(x) = 1$. So in that case, $\Lambda(x) = x$ is the only valid solution to the Key Equation, resulting in one error to be corrected.

Having taken care of the case $S^{-1}(x) \equiv x \mod g(x)$, let us now assume that this is not the case and continue. As $g(x)$ is irreducible, $\mathbb{F}_{2^m}[x]/\langle g(x) \rangle$ is a finite field of characteristic 2, so we can compute a square-root; in particular, we can find an $\tilde{S}(x)$ such that $\tilde{S}^2(x) \equiv x + S^{-1}(x) \mod g(x)$ and $\Delta \tilde{S} < \Delta g$. This value is directly computable by the receiver after having computed $S(x)$. Inserting $\tilde{S}(x)$ in (9), we get

$$\begin{aligned} b^2(x) \tilde{S}^2(x) \equiv a^2(x) \mod g(x) \iff \\ b(x) \tilde{S}(x) \equiv a(x) \mod g(x) \end{aligned} \tag{10}$$

Now we are in the case of a new Key Equation, where the degrees of the unknown polynomials are halved! We proceed in a manner resembling that of the GRS codes from the preceding section. The above equation tells us that $a(x) - y b(x) \in M = [g(x), y - \tilde{S}(x)]$. If we run the EA on $g(x)$ and $\tilde{S}(x)$, by Proposition 5, we get a Gröbner basis $G = \{h_1, h_2\}$ of M of module term order $<_\mu$ for any integer $\mu \geq 0$; for reasons becoming apparent momentarily, we choose $\mu = 1$.

By Proposition 2, we know there exist polynomials $f_1, f_2 \in \mathbb{F}[x]$ such that

$$a(x) - y b(x) = f_1(x) h_1(x, y) + f_2(x) h_2(x, y) \tag{11}$$

Remembering Proposition 1, assume that $\Delta^y_{<_1} h_2 = 1$ and therefore that $\Delta^y_{<_1} h_1 = 0$. Now, the case here is slightly more complicated than that of the GRS codes, as we do not know a priori which of $a(x)$ and $b(x)$ has the largest degree. If ϵ is even then $\Delta a = \frac{\epsilon}{2}$ and $\Delta b \leq \frac{\epsilon}{2} - 1$ whereby $a(x) >_1 y b(x)$. From Proposition 2 we then get

$$\begin{aligned} \Delta f_1 &= \Delta a - \Delta^x_{<_1}(h_1) &&= \tfrac{\epsilon}{2} - \Delta g + \Delta^x_{<_1}(h_2) \\ \Delta f_2 &\leq \Delta a - 1 - \Delta^x_{<_1}(h_2) &&= \tfrac{\epsilon}{2} - 1 - \Delta^x_{<_1}(h_2) \end{aligned} \tag{12}$$

In a similar manner, when ϵ is odd we get $a(x) <_1 y b(x)$ and

$$\begin{aligned} \Delta f_1 &\leq \Delta b + 1 - \Delta^x_{<_1}(h_1) - 1 &&= \tfrac{\epsilon-1}{2} - \Delta g + \Delta^x_{<_1}(h_2) \\ \Delta f_2 &= \Delta b - \Delta^x_{<_1}(h_2) &&= \tfrac{\epsilon-1}{2} - \Delta^x_{<_1}(h_2) \end{aligned} \tag{13}$$

In either of the above cases, we see that if $\epsilon \leq \Delta g$, one of the bounds for Δf_1 and Δf_2 will be negative, in which case either f_1 or f_2 will be zero. This in turn means that $a(x) - y b(x)$ will be a multiple of either h_1 or h_2, namely the one which has the same y-degree as $a(x) - y b(x)$ under $<_1$. As $\Lambda(x)$ is square-free, $a(x)$ and $b(x)$ must be relatively prime, so this multiple must be a constant. This corresponds to Patterson's decoder [7], except that there the BMA is used instead of the EA to solve (10). This requires an initial transformation of (10), and an "inverse" transformation on the output of the BMA.

In case f_1 and f_2 are both non-zero, spurred on by the success of the last section, we would like to be able to find them using rational interpolation. However, in the last section, we knew that the evaluation of the target polynomial $\Lambda(x)$ would be 0 in at least ϵ positions; for neither $a(x)$ nor $b(x)$ do we have such information. We therefore first need to re-enter (11) into their defining expression: $\Lambda(x) = a^2(x) + xb^2(x)$. Let first $h_1(x,y) = h_{10}(x) + yh_{11}(x)$ and $h_2(x,y) = h_{20}(x) + yh_{21}(x)$. Then using (11), we get

$$\begin{aligned} \Lambda(x) = \quad & (f_1(x)h_{10}(x) + f_2(x)h_{20}(x))^2 \\ & +x(f_1(x)h_{11}(x) + f_2(x)h_{21}(x))^2 \\ = & f_1^2(x)(h_{10}^2(x) + xh_{11}^2(x)) + f_2^2(x)(h_{20}^2(x) + xh_{21}^2(x)) \end{aligned}$$

Similarly to the preceding section, for at least ϵ values of $x_0 \in L$, we now know that $\Lambda(x_0) = 0$. For these ϵ values of x_0, by the above, we therefore have

$$f_1(x_0)\sqrt{\hat{h}_1(x_0)} + f_2(x_0)\sqrt{\hat{h}_2(x_0)} = 0$$

where $\hat{h}_1(x) = h_{10}^2(x) + xh_{11}^2(x)$ and $\hat{h}_2(x) = h_{20}^2(x) + xh_{21}^2(x)$. For us to be able to use Theorem 6, we have then only to certify that f_1 and f_2 are coprime, and that $\hat{h}_1$ and $\hat{h}_2$ will never simultaneously evaluate to zero. But the former is true since a and b are coprime which is due to Λ being square-free, and the latter is true since $h_1(x,y)$ and $h_2(x,y)$ are coprime. We have therefore finally arrived at a rational interpolation problem.

We will again use the results of Section III to solve this problem for some values of $\epsilon, n, \Delta g$. The next section is concerned with that analysis. The complete list decoder is shown in Algorithm 2.

Remark: As mentioned, Patterson's original algorithm [7] solves (10) using the BMA. One could possibly also extend this for list decoding using rational interpolation. However, a transformation is needed for letting the BMA solve (10), and this makes the details for rational interpolation less straightforward. One should also note that the BMA and the EA in their straightforward implementations have the same asymptotic running time $O(\theta^2)$ (see e.g. [5]), and that both admit a recursive version with asymptotic running time $O(\theta \log^2 \theta)$, where θ is the degree of the ingoing polynomials (see e.g. [30, Chapter 11.7] respectively [31, Chapter 8.9]). ■

C. Analysis of the parameters

For a given decoding radius τ, we want to know whether we can construct a $Q(x,y,z)$ such that whenever $\epsilon \le \tau$, we can find f_1 and f_2 in the manner specified in Theorem 6, and we want values for the parameters of ℓ and s.

We should set w_1, w_2 inspired by (12) and (13), but we need just one set of values which will cover both the even and odd cases. Therefore, we use for both f_1 and f_2 the larger of the degree bounds:

$$\begin{aligned} w_1 &= \tfrac{\tau}{2} - \Delta g + \Delta^x_{<_1}(h_2) \\ w_2 &= \tfrac{\tau-1}{2} - \Delta^x_{<_1}(h_2) \end{aligned} \tag{14}$$

Now define $w = w_1 + w_2 = \tau - \Delta g - \frac{1}{2}$. Note that w and either w_1 or w_2 will not be integer. Inserting the value for w and rearranging, (2) becomes

$$\frac{\tau}{n} < \frac{1}{(\ell+1)(\ell-2s)}\left(\binom{\ell+1}{2}\frac{\Delta g + \frac{1}{2}}{n} - 2\binom{s+1}{2}\right) \tag{15}$$

if we assume that $\ell > 2s$. Just as we before found that the governing equation for Algorithm 1 is parallel to that of the GSA, the above equation is parallel to the governing equation of the GSA+KV: using e.g. [19, Lemma 9.7] and setting the two multiplicities as $r = \ell - s$ and $\bar{r} = s$ we achieve the same equation. This means that Algorithm 2 has the same decoding radius as the GSA+KV when the choice of the two multiplicities are restricted thusly. From [19, Problem 9.9], the choice $\bar{r} = \ell - r$ exactly maximises the decoding radius which is then given in [19, Problem 9.10]. We also get $\bar{r} < r$ so $\bar{r} < \ell/2$ and hence in our case $\ell > 2s$; this is also what we assumed at (15) which means we can indeed reuse the analysis from the GSA+KV.

Lemma 16. *Algorithm 2 can list decode for any* $\tau < \frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\Delta g - 2)}$.

Proof: With the above duality, we have already established that for any given τ less than the given decoding radius we can select values of s and ℓ such that the sought f_1 and f_2 can be found whenever $\epsilon = \tau$. We again have to employ Lemma 7 in order to guarantee that f_1 and f_2 will be found when $\epsilon < \tau$. The lemma promises this if we can satisfy

$$\min\{w_1 - \Delta f_1, w_2 - \Delta f_2\} \ge \tfrac{s}{\ell}(\tau - \epsilon)$$

Using (12), (13) and (14), we see that $w_1 - \Delta f_1 \ge \frac{\tau}{2} - \lfloor\frac{\epsilon}{2}\rfloor \ge \frac{1}{2}(\tau - \epsilon)$, both when ϵ is even and when it's odd. Similarly for $w_2 - \Delta f_2$. The condition of Lemma 7 is then always satisfied as long as $\ell > 2s$. This we already assumed at (15). □

Remark: It is the necessity of having to use Lemma 7 that adds the peculiar complication on the setting of w_1 and w_2. If we choose a τ, we will know its parity, so we could choose w_2 and w_2 from (12) or (13), according to that parity. This would allow us to decode exactly τ errors; analysis shows that in that case one could choose any $\tau < \frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\Delta g - 4)}$, i.e. slightly greater than the binary Johnson radius. However, the condition of Lemma 7 would then not always be true so we would not always be able to correct *fewer* errors. This is the reason of having to set w_1 and w_2 as in (14).

Interestingly, if we allow two runs of the rational interpolation procedure instead of just one, we *can* achieve the decoding radius $\tau < \frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\Delta g - 4)}$ and still also decode fewer than τ errors: let the first run be responsible for finding those error locators corresponding to even number of errors, and the second run for the odd number of errors. For each run we then only need a looser version of Lemma 7, where only a number of points with the right parity need to be interpolated as well. Then we can set w_1, w_2 according to (12) in the even-parity run, and similarly w_1, w_2 from (13) in the odd-parity run. This yields the mentioned decoding radius. ■

Lemma 17. *For given n, Δg and τ, then ℓ and s are valid choices for the parameters for Algorithm 2 if and only if ℓ, $r = \ell - s$ and $\bar{r} = s$ are valid parameters for the GSA+KV as described in [19, §9.6].*

For closed expressions for valid values of the parameters ℓ and s, one can use the analysis of Trifonov and Lee [8] which works for any application of the rational interpolation method.

Algorithm 2 Wu list decoding binary Goppa codes

```
Input: A binary Goppa code $\mathcal{C}$ with Goppa polynomial
    $g(x) \in \mathbb{F}_{2^m}[x]$ and evaluation points $\alpha_0, \dots, \alpha_{n-1}$, a
    decoding radius $\tau < \frac{1}{2}n - \frac{1}{2}\sqrt{n - 4\Delta g - 2}$, and a
    received word $r \in \mathbb{F}_2^n$.
Output: A list of all codewords in $\mathcal{C}$ within radius $\tau$ of $r$ or
    Fail if there are no such words.

1: Calculate the syndrome $S(x)$ from $r$ according to (7). If
   $S^{-1}(x) = x$ and $0 \in L$, then flip the corresponding bit of
   $r$ and return that word. If $S^{-1}(x) = x$ and $0 \notin L$, return
   Fail. Otherwise, calculate $\tilde{S}(x)$ satisfying $\Delta\tilde{S} < \Delta g$ and
   $\tilde{S}^2(x) \equiv x + S^{-1}(x) \mod g(x)$.
2: Run the EA on $g(x), \tilde{S}(x)$ and halt when $\Delta s_i < \Delta v_i + 1$,
   reusing the notation of Section II. Define $\hat{h}_1(x) =$
   $s_{i-1}^2(x) + x v_{i-1}^2(x)$ and $\hat{h}_2(x) = s_i^2(x) + x v_i^2(x)$.
3: If either $\hat{h}_1(x)$ or $\hat{h}_2(x)$ are valid error-locators of degree
   at most $2\Delta g - \tau$, use that to decode, and if this yields a
   word in $\mathcal{C}$, return this one word.
4: Otherwise, we seek $f_1, f_2$ according to (11). Set $w_1, w_2$ as
   in (14), and calculate $\ell$ and $s$ to satisfy (15). Construct a
   $Q(x, y, z)$ satisfying the requirements of Theorem 6 using
   the points $\{(\alpha_i, \sqrt{\hat{h}_1(\alpha_i)}, \sqrt{\hat{h}_2(\alpha_i)})\}_{i=0}^{n-1}$.
5: Find all factors of $Q(x, y, z)$ of the form $y f_1^\star(x) + z f_2^\star(x)$
   where $f_1^\star$ and $f_2^\star$ have degree less than $w_1$ and $w_2$
   respectively. Return Fail if no such factors exist.
6: For each such factor, construct $\Lambda^\star(x) = f_1^{\star 2}(x)\hat{h}_1(x) +$
   $f_2^{\star 2}(x)\hat{h}_2(x)$. If it is a valid error-locator, use it for
   decoding $r$. Return Fail if none of the factors yield error-
   locators
7: Return those of the decoded words that are in $\mathcal{C}$. Return
   Fail if there are no such words.
```

D. Complexity Analysis

Again, the complexity of Algorithm 2 is easily found using the results of Section III-A. For simplicity, we will assume that $2^m \in O(n)$. In that case, as $\ell \geq s$, steps 4 and 5 can be computed in $O(\ell^\omega s n \log^{O(1)}(\ell n))$. The remaining steps are of lower order, seen using arguments similar to those in Section IV-D.

Lemma 18. *If $2^m \in O(n)$ then Algorithm 2 has complexity $O(\ell^\omega s n \log^{O(1)}(\ell n))$.*

The GSA+KV can decode binary Goppa codes – in fact any alternant code – up to the small-field Johnson bound. Also here, the bottle-neck of the complexity is the construction of the interpolation polynomial. Bernstein in [15] gives an algorithm for constructing this fast, and in terms of ℓ and n and relaxing s, r and $\bar{r}$ to ℓ, it has the same complexity as the above.

However, similarly to Section IV-D, one should note that $s = \bar{r} < \ell/2$ and $r = \ell - \bar{r} > \ell/2$, and the difference between s and r increases with the rate of the code. From this view, one would therefore expect that Algorithm 2 outperforms the GSA+KV, though the asymptotic analysis we have performed here is too crude to say for certain.

VI. Conclusion

In this article, we have reinvestigated the Wu list decoder of [1]. Originally formulated in tight integration with the Berlekamp-Massey algorithm, we have shown how the extended Euclidean algorithm can be used instead, enabling one to solve more general equations than the original Key Equation for Generalised Reed-Solomon codes.

At its core, the Wu list decoder solves a rational interpolation problem in a manner mirroring the polynomial interpolation of the Guruswami-Sudan algorithm (GSA). We have pointed out how this equation becomes the one of the GSA by a change of variables, implying that their decoding radii and list sizes are the same, as well as connecting the multiplicities.

The most expensive part of solving the rational interpolation problem is the construction of an interpolation polynomial. We have shown how to extend methods used in the GSA for constructing this polynomial fast. The result is that the Wu list decoder can be made to run in the same complexity as the fastest variants of the GSA.

The decoupling of the Key Equation-solving and rational interpolation from the actual decoding results in a short derivation of the list decoder for GRS codes. Moreover, it makes it clear that the approach also can be used to extend the Patterson decoder for binary Goppa codes, list decoding up to the binary Johnson radius. Also here, a connection to the governing equation of the GSA with the Kötter-Vardy multiplicity assignment method is pointed out.

VII. Acknowledgements

The authors are very grateful for the insightful comments and suggestions made by the anonymous reviewers. The authors gratefully acknowledge the support from the Danish National Research Foundation and the National Science Foundation of China (Grant No.11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography.

References

[1] Y. Wu, "New List Decoding Algorithms for Reed-Solomon and BCH Codes," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3611–3630, 2008.

[2] V. Guruswami and M. Sudan, “Improved Decoding of Reed-Solomon Codes and Algebraic Geometry Codes,” *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1757–1767, 1999.

[3] E. R. Berlekamp, *Algebraic Coding Theory*. Aegean Park Press, 1968.

[4] P. Fitzpatrick, “On the Key Equation,” *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1290–1302, 1995.

[5] J. Dornstetter, “On the Equivalence Between Berlekamp’s and Euclid’s Algorithms,” *IEEE Transactions on Information Theory*, vol. 33, no. 3, pp. 428–431, 1987.

[6] A. E. Heydtmann and J. M. Jensen, “On the Equivalence of the Berlekamp-Massey and the Euclidean Algorithms for Decoding,” *IEEE Transactions on Information Theory*, vol. 46, no. 7, pp. 2614–2624, 2000.

[7] N. Patterson, “The Algebraic Decoding of Goppa Codes,” *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 203–207, 1975.

[8] P. Trifonov and M. Lee, “Efficient Interpolation in the Wu List Decoding Algorithm,” *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 5963–5971, 2012.

[9] P. V. Trifonov, “Another Derivation of Wu List Decoding Algorithm and Interpolation in Rational Curve Fitting,” in *Proc. of IEEE R8 SIBIRCON*, 2010, pp. 59–64.

[10] M. Ali and M. Kuijper, “A Parametric Approach to List Decoding of Reed-Solomon Codes Using Interpolation,” *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6718–6728, 2011.

[11] P. Beelen and K. Brander, “Key-equations for list decoding of Reed-Solomon codes and how to solve them,” *Journal of Symbolic Computation*, vol. 45, no. 7, pp. 773–786, 2010.

[12] K. Lee and M. E. O’Sullivan, “List Decoding of Reed-Solomon Codes from a Gröbner Basis Perspective,” *Journal of Symbolic Computation*, vol. 43, no. 9, pp. 645 – 658, 2008.

[13] M. Alekhnovich, “Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed-Solomon Codes,” *IEEE Transactions on Information Theory*, vol. 51, no. 7, 2005.

[14] P. Giorgi, C. Jeannerod, and G. Villard, “On the Complexity of Polynomial Matrix Computations,” in *Proceedings of International Symposium on Symbolic and Algebraic Computation ’03*. ACM, 2003, pp. 135–142.

[15] D. Bernstein, “Simplified high-speed high-distance list decoding for alternant codes,” 2011. [Online]. Available: http://cr.yp.to/papers.html#simplelist

[16] F. J. MacWilliams and N. J. Sloane, *The Theory of Error-Correcting Codes*. North-Holland Publishing, 1977.

[17] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, “Further Results on Goppa Codes and Their Applications to Constructing Efficient Binary Codes,” *IEEE Transactions on Information Theory*, vol. 22, no. 5, pp. 518–526, 1976.

[18] D. Augot, M. Barbier, and A. Couvreur, “List-decoding of binary Goppa codes up to the binary Johnson bound,” *arXiv*, vol. abs/1012.3439, 2010. [Online]. Available: http://arxiv.org/abs/1012.3439v1

[19] R. Roth, *Introduction to Coding Theory*. Cambridge University Press, 2006.

[20] D. Cox, J. Little, and D. O’Shea, *Using Algebraic Geometry*. Springer Verlag, 1998, vol. 185.

[21] H. C. A. van Tilborg, *Error-Correcting Codes - A First Course*. Studentlitteratur, 1993.

[22] A. Lenstra, “Factoring Multivariate Polynomials over Finite Fields,” *Journal of Computer and System Sciences*, vol. 30, no. 2, pp. 235–248, 1985.

[23] T. Mulders and A. Storjohann, “On lattice reduction for polynomial matrices,” *Journal of Symbolic Computation*, vol. 35, no. 4, pp. 377–401, 2003.

[24] D. Coppersmith and S. Winograd, “Matrix Multiplication via Arithmetic Progressions,” *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 251–280, 1990.

[25] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge Univ Press, 2003.

[26] R. Roth and G. Ruckenstein, “Efficient Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance,” *IEEE Transactions on Information Theory*, vol. 46, no. 1, pp. 246 –257, 2000.

[27] A. Zeh, C. Gentner, and D. Augot, “An Interpolation Procedure for List Decoding Reed-Solomon Codes Based on Generalized Key Equations,” *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 5946–5959, 2011.

[28] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, “A Method for Solving Key Equation for Decoding Goppa Codes,” *Information and Control*, vol. 27, no. 1, pp. 87–99, 1975.

[29] R. McEliece, “The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes,” *IPN progress report*, pp. 42–153, 2003.

[30] R. Blahut, *Theory and Practice of Error Control Codes*. Addison-Wesley, 1983.

[31] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis Of Computer Algorithms*. Addison-Wesley, 1974.