Multi-Trial Guruswami–Sudan Decoding for Generalised Reed–Solomon Codes

Johan Sebastian Rosenkilde Nielsen, Alexander Zeh

Noname manuscript No.
(will be inserted by the editor)

Multi-Trial Guruswami–Sudan Decoding for Generalised Reed–Solomon Codes

Johan S. R. Nielsen · Alexander Zeh

April 9, 2014

Abstract An iterated refinement procedure for the Guruswami–Sudan list decoding algorithm for Generalised Reed–Solomon codes based on Alekhnovich's module minimisation is proposed. The method is parametrisable and allows variants of the usual list decoding approach. In particular, finding the list of *closest* codewords within an intermediate radius can be performed with improved average-case complexity while retaining the worst-case complexity.

We provide a detailed description of the module minimisation, reanalysing the Mulders–Storjohann algorithm and drawing new connections to both Alekhnovich's algorithm and Lee–O'Sullivan's. Furthermore, we show how to incorporate the re-encoding technique of Kötter and Vardy into our iterative algorithm.

Keywords Guruswami–Sudan · List Decoding · Multi-Trial · Reed–Solomon Codes · Re-Encoding Transformation

1 Introduction

Since the discovery of a polynomial-time hard-decision list decoder for Generalised Reed–Solomon (GRS) codes by Guruswami and Sudan (GS) [23, 11] in the late 1990s, much work has been done to speed up the two main parts of the algorithm: interpolation and root-finding. Notably, for interpolation Beelen and Brander [3] mixed the module reduction approach by Lee and O'Sullivan [16] with the parametrisation of Zeh *et al.* [25], and employed the fast module reduction algorithm by Alekhnovich [1]. Bernstein [5] pointed out that an asymptotically faster variant can be achieved by using the reduction algorithm by Giorgi *et al.* [10]. Very recently Chowdhury *et al.* [8] used fast displacement-rank linear-algebraic solvers to achieve the fastest known approach. The

J. S. R. Nielsen is with the Institute of Communications Engineering, University of Ulm, Germany.
E-mail: jsrn@jsrn.dk
A. Zeh was with the Institute of Communications Engineering, University of Ulm, Ulm, Germany and Research Center INRIA Saclay - Île-de-France, École Polytechnique, France and is now with the Computer Science Department, Technion—Israel Institute of Technology, Haifa, Israel.
E-mail: alex@codingtheory.eu

GS approach was generalised by Kötter and Vardy to a soft-decision scenario [15], and the same authors also presented a complexity-reducing re-encoding transformation [14, 13]. Cassuto *et al.* [7] and Tang *et al.* [24] proposed modified interpolation-based list decoders with reduced average-complexity.

For the root-finding step, one can employ the method of Roth and Ruckenstein [21] in a divide-and-conquer fashion, as described by Alekhnovich [1]. This step then becomes an order of magnitude faster than interpolation, leaving the latter as the main target for further optimisations.

For a given GRS code, the GS algorithm has two parameters, both positive integers: the interpolation multiplicity s and the list size ℓ. Together with the code parameters they determine the decoding radius τ. To achieve a higher decoding radius for some given GRS code, one needs higher s and ℓ, and the value of these strongly influences the running time of the algorithm.

In this work, we present a novel iterative method: we first solve the interpolation problem for $s = \ell = 1$ and then iteratively refine this solution for increasing s and ℓ. In each step of our algorithm, we obtain a valid solution to the interpolation problem for these intermediate parameters. The method builds upon that of Beelen–Brander [3], but a new analysis of the computational engine—Alekhnovich's module minimisation algorithm—reveals that each iteration runs faster than otherwise expected.

The method therefore allows a fast *multi-trial* list decoder when our aim is just to find the list of codewords with minimal distance to the received word. At any time during the refinement process, we will have an interpolation polynomial for intermediate parameters $\hat{s} \leq s$, $\hat{\ell} \leq \ell$ yielding an intermediate decoding radius $d/2 \leq \hat{\tau} \leq \tau$, where d is the minimum distance. If we perform the root-finding step of the GS algorithm on this, all codewords with distance at most $\hat{\tau}$ from the received are returned; if there are any such words, we break computation and return those; otherwise we continue the refinement. We can choose any number of these trials, e.g. for each possible intermediate decoding radius between $d/2$ and the target τ.

Since the root-finding step of GS is less complex than the interpolation step, this multi-trial decoder will have the same asymptotic worst-case complexity as the usual GS using the Beelen–Brander interpolation [3]. However, its average-case complexity is better since due to the properties of the underlying channel it is more probable to have a small number of errors rather than a big one.

This contribution is structured as follows. In the next section we give necessary preliminaries and state the GS interpolation problem for decoding GRS codes. In Section 3 we give a definition and properties of minimal matrices. We describe and reanalyse the conceptually simple Mulders–Storjohann algorithm [18] for bringing matrices to this form. We also give a fresh look at Alekhnovich's algorithm [1] simply as a divide-&-conquer variant of Mulders–Storjohann, and our new analysis can carry over. Our new iterative procedure is explained in detail in Section 4 and the incorporation of the re-encoding transformation [13] is described in Section 5. In Section 6 we present simulation results.

Parts of these results were presented at WCC 2013 [20]; compared to that article, this version contains the incorporation of the re-encoding scheme, a full example of the algorithm, simulation results, as well as a more detailed description of the module minimisation procedure.

2 Preliminaries

2.1 Notation

Let $\mathbb{F}_q$ be the finite field of order q and let $\mathbb{F}_q[X]$ be the polynomial ring over $\mathbb{F}_q$ with indeterminate X. Let $\mathbb{F}_q[X,Y]$ denote the polynomial ring in the variables X and Y and let $\operatorname{wdeg}_{u,v} X^iY^j \triangleq ui + vj$ be the (u,v)-weighted degree of X^iY^j.

A vector of length n is denoted by $\mathbf{v} = (v_0, \ldots, v_{n-1})$. If $\mathbf{v}$ is a vector over $\mathbb{F}_q[X]$, let $\deg \mathbf{v} \triangleq \max_i\{\deg v_i(X)\}$. We introduce the leading position as

$$\operatorname{LP}(\mathbf{v}) = \max_i \big\{ i \mid \deg v_i(X) = \deg \mathbf{v} \big\} \tag{1}$$

and the leading term $\operatorname{LT}(\mathbf{v}) = v_{\operatorname{LP}(\mathbf{v})}$ is the term at this position. An $m \times n$ matrix is denoted by $\mathcal{V} = \|v_{i,j}\|_{i=0,j=0}^{m-1,n-1}$. The rows of such a matrix are denoted by bold lower-case letters, e.g. $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$. Furthermore, let the degree of such a polynomial matrix be $\deg \mathcal{V} = \sum_{i=0}^{m-1} \deg \mathbf{v}_i$. Modules are denoted by capital letters such as M.

2.2 Interpolation-Based Decoding of GRS Codes

Let $\alpha_0, \ldots, \alpha_{n-1}$ be n nonzero distinct elements of $\mathbb{F}_q$ with $n < q$ and let $w_0, \ldots, w_{n-1}$ be n (not necessarily distinct) nonzero elements of $\mathbb{F}_q$. A GRS code $\mathcal{GRS}(n,k)$ of length n and dimension k over $\mathbb{F}_q$ is given by

$$\mathcal{GRS}(n,k) \triangleq \big\{ (w_0 f(\alpha_0), \ldots, w_{n-1} f(\alpha_{n-1})) : f(X) \in \mathbb{F}_q[X],\, \deg f(X) < k \big\}. \tag{2}$$

GRS codes are Maximum Distance Separable (MDS) codes, i.e., their minimum Hamming distance is $d = n - k + 1$. We shortly explain the interpolation problem of GS [11, 23] for list decoding GRS codes up to the Johnson radius [12,2] in the following.

Theorem 1 (Guruswami–Sudan for GRS Codes [11,23]) *Let $\mathbf{c} \in \mathcal{GRS}(n,k)$ and $f(X)$ be the corresponding information polynomial as defined in (2). Let $\mathbf{r} = (r_0, \ldots, r_{n-1}) = \mathbf{c} + \mathbf{e}$ be a received word where $\operatorname{weight}(\mathbf{e}) \le \tau$. Let r_i' denote r_i/w_i.*

Let $Q(X,Y) \in \mathbb{F}_q[X,Y]$ be a nonzero polynomial that passes through the n points $(\alpha_0, r_0'), \ldots, (\alpha_{n-1}, r_{n-1}')$ with multiplicity $s \ge 1$, has Y-degree at most ℓ, and $\operatorname{wdeg}_{1,k-1} Q(X,Y) < s(n-\tau)$. Then $(Y - f(X)) \mid Q(X,Y)$.

One can easily show that a polynomial $Q(X,Y)$ that fulfills the above conditions can be constructed whenever $E(s,\ell,\tau) > 0$, where

$$E(s,\ell,\tau) \triangleq (\ell+1)s(n-\tau) - \tbinom{\ell+1}{2}(k-1) - \tbinom{s+1}{2}n \tag{3}$$

is the difference between the maximal number of coefficients of $Q(X,Y)$, and the number of homogeneous linear equations on $Q(X,Y)$ specified by the interpolation constraint. This determines the maximal number of correctable errors, and one can show that satisfactory s and ℓ can always be chosen whenever $\tau < n - \sqrt{n(k-1)}$.

Definition 2 (Permissible Triples) *An integer triple $(s,\ell,\tau) \in (\mathbb{Z}_+)^3$ is* permissible *if $E(s,\ell,\tau) > 0$.*

We define also the decoding radius-function $\tau(s,\ell)$ as the greatest integer such that $(s,\ell,\tau(s,\ell))$ is permissible.

It is well-known that $E(s,\ell,\tau) > 0$ for $s > \ell$ implies $\tau < d/2$, which is half the minimum distance. Therefore, it never makes sense to consider $s > \ell$, and in the remainder we will always assume $s \le \ell$. Furthermore, we will also assume $s, \ell \in O(n^2)$ since this e.g. holds for any τ for the closed-form expressions in [11].

Let us illustrate the above. The following will be a running example throughout the article.

Example 3 *A $\mathcal{GRS}(16,4)$ code over $\mathbb{F}_{17}$ can uniquely correct $\tau_0 = (n-k)/2 = 6$ errors; unique decoding corresponds to $s_0 = \ell_0 = 1$ and one can confirm that $E(1,1,6) > 0$. To attain a decoding radius $\tau_1 = 7$, one can choose $s_1 = 1$ and $\ell_1 = 2$ in order to obtain a permissible triple. Also $(1,3,7)$ is permissible, though less interesting since it does not give improved decoding radius. However, one finds $(2,4,8)$ and $(28,64,9)$ are permissible. Since $n - \sqrt{n(k-1)} < 10$, there are no permissible triples for greater decoding radii.*

2.3 Module Reformulation of Guruswami–Sudan

Let $M_{s,\ell} \subset \mathbb{F}_q[X,Y]$ denote the space of all bivariate polynomials passing through the points $(\alpha_0, r'_0), \ldots, (\alpha_{n-1}, r'_{n-1})$ with multiplicity s and Y-degree at most ℓ. We are searching for an element of $M_{s,\ell}$ with low $(1,k-1)$-weighted degree.

Following the ideas of Lee and O'Sullivan [16], we can first remark that $M_{s,\ell}$ is an $\mathbb{F}_q[X]$-module. Second, we can give an explicit basis for $M_{s,\ell}$. Define first two polynomials $G(X) \triangleq \prod_{i=0}^{n-1}(X-\alpha_i)$ as well as $R(X)$ in $\mathbb{F}_q[X]$ as the unique Lagrange interpolation polynomial going through the points (α_i, r'_i) for $i = 0, \ldots, n-1$. Denote by $Q_{[t]}(X)$ the Y^t-coefficient of $Q(X,Y)$ when Q is regarded over $\mathbb{F}_q[X][Y]$.

Lemma 4 *Let $Q(X,Y) = \sum_{i=0}^{t} Q_{[i]}(X)Y^i \in M_{s,\ell}$ with $\operatorname{wdeg}_{0,1} Q = t < s$. Then $G(X)^{s-t} \mid Q_{[t]}(X)$.*

Proof $Q(X,Y)$ interpolates the n points (α_i, r'_i) with multiplicity s, so for any i, $Q(X+\alpha_i, Y+r'_i) = \sum_{j=0}^{t} Q_{[j]}(X+\alpha_j)(Y+r'_j)^j$ has no monomials of total degree less than s. Multiplying out the $(Y+r'_j)^j$-terms, $Q_{[t]}(X+\alpha_j)Y^t$ is the only term with Y-degree t. Therefore $Q_{[t]}(X+\alpha_j)$ can have no monomials of degree less than $s-t$, which implies $(X-\alpha_i)^{s-t} \mid Q_{[t]}(X)$. As this holds for any i, we proved the lemma. □

Theorem 5 *The module $M_{s,\ell}$ is generated as an $\mathbb{F}_q[X]$-module by the $\ell+1$ polynomials $P^{(i)}(X,Y) \in \mathbb{F}_q[X,Y]$ given by*

$$P^{(t)}(X,Y) = G(X)^{s-t}(Y - R(X))^t, \qquad \text{for } 0 \le t < s,$$
$$P^{(t)}(X,Y) = Y^{t-s}(Y - R(X))^s, \qquad \text{for } s \le t \le \ell.$$

Proof It is easy to see that each $P^{(t)}(X,Y) \in M_{s,\ell}$ since both $G(X)$ and $(Y-R(X))$ go through the n points (α_i, r'_i) with multiplicity one, and that $G(X)$ and $(Y-R(X))$ divide $P^{(t)}(X,Y)$ with total power s for each t.

To see that any element of $M_{s,\ell}$ can be written as an $\mathbb{F}_q[X]$-combination of the $P^{(t)}(X,Y)$, let $Q(X,Y)$ be some element of $M_{s,\ell}$. Then the polynomial $Q^{(\ell-1)}(X,Y) = Q(X,Y) - Q_{[\ell]}(X)P^{(\ell)}(X,Y)$ has Y-degree at most $\ell-1$. Since both $Q(X,Y)$ and $P^{(\ell)}(X,Y)$ are in $M_{s,\ell}$, so must $Q^{(\ell-1)}(X,Y)$ be in $M_{s,\ell}$. Since $P^{(t)}(X,Y)$ has Y-degree t and $P^{(t)}_{[t]}(X) = 1$ for $t = \ell, \ell-1, \ldots, s$, we can continue reducing this way until we reach a $Q^{(s-1)}(X,Y) \in M_{s,\ell}$ with Y-degree at most $s-1$. From then on, we have $P^{(t)}_{[t]}(X) = G(X)^{s-t}$, but by Lemma 4, we must also have $G(X) \mid Q^{(s-1)}_{[s-1]}(X)$. Therefore, we can reduce by $P^{(s-1)}(X,Y)$. This can be continued with the remaining $P^{(t)}(X,Y)$, eventually reducing the remainder to 0. □

We can represent the basis of $M_{s,\ell}$ by the $(\ell+1)\times(\ell+1)$ matrix $\mathcal{A}_{s,\ell} = \|P^{(i)}_{[j]}(X)\|^{\ell,\ell}_{i=0,j=0}$ over $\mathbb{F}_q[X]$; more explicitly we have:

$$\mathcal{A}_{s,\ell} \triangleq \begin{pmatrix} G^s & & & & & \\ G^{s-1}(-R) & G^{s-1} & & & & \\ G^{s-2}(-R)^2 & 2G^{s-2}(-R) & G^{s-2} & & 0 & \\ \vdots & & & \ddots & & \\ (-R)^s & \binom{s}{1}(-R)^{s-1} & \ldots & 1 & & \\ & (-R)^s & & \ldots & 1 & \\ & & \ddots & & & \ddots \\ 0 & & & (-R)^s & \ldots & 1 \end{pmatrix}. \tag{4}$$

Any $\mathbb{F}_q[X]$-linear combination of rows of $\mathcal{A}_{s,\ell}$ thus corresponds to an element in $M_{s,\ell}$ by its tth position being the $\mathbb{F}_q[X]$-coefficient to Y^t. All other bases of $M_{s,\ell}$ can similarly be represented by matrices, and these will be unimodular equivalent to $\mathcal{A}_{s,\ell}$, i.e., they can be obtained by multiplying $\mathcal{A}_{s,\ell}$ on the left with an invertible matrix over $\mathbb{F}_q[X]$.

Extending the work of Lee and O'Sullivan [16], Beelen and Brander [3] gave a fast algorithm for computing a satisfactory $Q(X,Y)$: start with $\mathcal{A}_{s,\ell}$ as a basis of $M_{s,\ell}$ and compute a different, "minimal" basis of $M_{s,\ell}$ where an element of minimal $(1,k-1)$-weighted degree appears directly.

In the following section, we give further details on how to compute such a basis, but our ultimate aim in Section 4 is different: we will use a minimal basis of $M_{s,\ell}$ to efficiently compute one for $M_{\hat{s},\hat{\ell}}$ for $\hat{s} \geq s$ and $\hat{\ell} > \ell$. This will allow an iterative refinement for increasing s and ℓ, where after each step we have such a minimal basis for $M_{s,\ell}$. We then exploit this added flexibility in our multi-trial algorithm.

3 Module Minimisation

Given a basis of $M_{s,\ell}$, e.g. $\mathcal{A}_{s,\ell}$, the module minimisation here refers to the process of obtaining a new basis, which is the smallest among all bases of $M_{s,\ell}$ in a precise sense. We will define this and connect various known properties of such matrices. We will then show how to perform this minimisation using the Mulders–Storjohann algorithm [18], reanalyse its performance and connect it to Alekhnovich's algorithm [1].

Definition 6 (Weak Popov Form [18]) *A matrix $\mathcal{V}$ over $\mathbb{F}_q[X]$ is in* weak Popov form *if the leading position of each row is different.*

We are essentially interested in short vectors in a module, and the following lemma shows that the simple concept of weak Popov form will provide this. It is a paraphrasing of [1, Proposition 2.3] and we omit the proof.

Lemma 7 (Minimal Degree) *If a square matrix $\mathcal{V}$ over $\mathbb{F}_q[X]$ is in weak Popov form, then one of its rows has minimal degree of all vectors in the row space of $\mathcal{V}$.*

Denote now by $\mathcal{W}_\ell$ the diagonal $(\ell+1)\times(\ell+1)$ matrix over $\mathbb{F}_q[X]$:

$$\mathcal{W}_\ell \triangleq \operatorname{diag}\left(1, X^{k-1}, \ldots, X^{\ell(k-1)}\right). \tag{5}$$

Since we seek a polynomial of minimal $(1, k-1)$-weighted degree, we also need the following corollary.

Corollary 8 (Minimal Weighted Degree) *Let $\mathcal{B} \in \mathbb{F}_q[X]^{(\ell+1)\times(\ell+1)}$ be the matrix representation of a basis of $M_{s,\ell}$. If $\mathcal{B}\mathcal{W}_\ell$ is in weak Popov form, then one of the rows of $\mathcal{B}$ corresponds to a polynomial in $M_{s,\ell}$ with minimal $(1, k-1)$-weighted degree.*

Proof Let $\widetilde{\mathcal{B}} = \mathcal{B}\mathcal{W}_\ell$. Now, $\widetilde{\mathcal{B}}$ will correspond to the basis of an $\mathbb{F}_q[X]$-module $\widetilde{M}$ isomorphic to $M_{s,\ell}$, where an element $Q(X,Y) \in M_{s,\ell}$ is mapped to $Q(X, X^{k-1}Y) \in \widetilde{M}$. By Lemma 7, the row of minimal degree in $\widetilde{\mathcal{B}}$ corresponds to an element of $\widetilde{M}$ with minimal X-degree. Therefore, the same row of $\mathcal{B}$ corresponds to an element of $M_{s,\ell}$ with minimal $(1, k-1)$-weighted degree. □

If for some matrix $\mathcal{B} \in \mathbb{F}_q[X]^{(\ell+1)\times(\ell+1)}$, $\mathcal{B}\mathcal{W}_\ell$ is in weak Popov form, we say that $\mathcal{B}$ is in *weighted weak Popov form.*

We introduce what will turn out to be a measure of how far a matrix is from being in weak Popov form.

Definition 9 (Orthogonality Defect [17]) *The* orthogonality defect *of a square matrix $\mathcal{V}$ over $\mathbb{F}_q[X]$ is defined as*

$$\mathrm{D}(\mathcal{V}) \triangleq \deg \mathcal{V} - \deg \det \mathcal{V}.$$

Lemma 10 *If a square matrix $\mathcal{V}$ over $\mathbb{F}_q[X]$ is in weak Popov form, then* $\mathrm{D}(\mathcal{V}) = 0$.

Proof Let $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$ be the rows of $\mathcal{V} \in \mathbb{F}_q[X]^{m\times m}$ and $\mathbf{v}_i = (v_{i,0}, \ldots, v_{i,m-1})$. In the alternating sum-expression for $\det \mathcal{V}$, the term $\prod_{i=0}^{m-1} \mathrm{LT}(\mathbf{v}_i)$ will occur since the leading positions of $\mathbf{v}_i$ are all different. Thus $\deg \det \mathcal{V} = \sum_{i=0}^{m-1} \deg \mathrm{LT}(\mathbf{v}_i) = \deg \mathcal{V}$ unless leading term cancellation occurs in the determinant expression. However, no other term in the determinant has this degree: regard some (unsigned) term in $\det \mathcal{V}$, say $t = \prod_{i=0}^{m-1} v_{i,\sigma(i)}$ for some permutation $\sigma \in S_m$. If not $\sigma(i) = \mathrm{LP}(\mathbf{v}_i)$ for all i (as defined in (1)), then there must be an i such that $\sigma(i) > \mathrm{LP}(\mathbf{v}_i)$ since $\sum_j \sigma(j)$ is the same for all $\sigma \in S_m$. Thus, $\deg v_{i,\sigma(i)} < \deg v_{i,\mathrm{LP}(\mathbf{v}_i)}$. As none of the other terms in t can have greater degree than their corresponding row's leading term, we get $\deg t < \sum_{i=0}^{m-1} \deg \mathrm{LT}(\mathbf{v}_i)$. Thus, $\mathrm{D}(\mathcal{V}) = 0$. □

Algorithm 1: Mulders-Storjohann [18]

Input: $\mathcal{V} \in \mathbb{F}_q[X]^{m\times m}$
Output: A matrix unimodular equivalent to $\mathcal{V}$ and in weak Popov form.

1 Apply row reductions as in Definition 11 on the rows of $\mathcal{V}$ until no longer possible
2 **return** this matrix.

Remark The weak Popov form is highly related to minimal Gröbner bases of the row space module, using a term order where vectors in $\mathbb{F}_q[X]^m$ are ordered according to their degree; indeed the rows of a matrix in weak Popov form *is* such a Gröbner basis (though the opposite is not always true). Similarly, the weighted weak Popov form has a corresponding weighted term order. In this light, Lemma 7 is simply the familiar assertion that a Gröbner basis over such a term order must contain a "minimal" element of the module. See e.g. [19, Chapter 2] for more details on this correspondence. The language of Gröbner bases was employed in the related works of [3,16].

3.1 Algorithms

Definition 11 (Row Reduction) *Applying a* row reduction *on a matrix over* $\mathbb{F}_q[X]$ *means to find two different rows* $\mathbf{v}_i, \mathbf{v}_j$, $\deg \mathbf{v}_i < \deg \mathbf{v}_j$ *and such that* $\mathrm{LP}(\mathbf{v}_i) = \mathrm{LP}(\mathbf{v}_j)$, *and then replacing* $\mathbf{v}_j$ *with* $\mathbf{v}_j - \alpha X^\delta \mathbf{v}_i$ *where* $\alpha \in \mathbb{F}_q$ *and* $\delta \in \mathbb{Z}_+$ *are chosen such that the leading term of the polynomial* $\mathrm{LT}(\mathbf{v}_j)$ *is cancelled.*

Algorithm 1 is due to Mulders and Storjohann [18]. Our proof of it is similar, though we have related the termination condition to the orthogonality defect, restricting it to only square matrices.

Introduce for the proof a *value function* $\psi : \mathbb{F}_q[X]^m \to \mathbb{N}_0$ as $\psi(\mathbf{v}) = m \deg \mathbf{v} + \mathrm{LP}(\mathbf{v})$. First let us consider the following lemma.

Lemma 12 *If we replace* $\mathbf{v}_j$ *with* $\mathbf{v}'_j$ *in a row reduction, then* $\psi(\mathbf{v}'_j) < \psi(\mathbf{v}_j)$.

Proof We cannot have $\deg \mathbf{v}'_j > \deg \mathbf{v}_j$ since all terms of both $\mathbf{v}_j$ and $\alpha X^\delta \mathbf{v}_i$ have degree at most $\deg \mathbf{v}_j$. If $\deg \mathbf{v}'_j < \deg \mathbf{v}_j$ we are done since $\mathrm{LP}(\mathbf{v}'_j) < m$, so assume $\deg \mathbf{v}'_j = \deg \mathbf{v}_j$. Let $h = \mathrm{LP}(\mathbf{v}_j) = \mathrm{LP}(\mathbf{v}_i)$. By the definition of the leading position, all terms in both $\mathbf{v}_j$ and $\alpha X^\delta \mathbf{v}_i$ to the right of h must have degree less than $\deg \mathbf{v}_j$, and so also all terms in $\mathbf{v}'_j$ to the right of h satisfies this. The row reduction ensures that $\deg v'_{j,h} < \deg v_{j,h}$, so it must then be the case that $\mathrm{LP}(\mathbf{v}'_j) < h$.

Theorem 13 *Algorithm 1 is correct. For a matrix* $\mathcal{V} \in \mathbb{F}_q[X]^{m\times m}$, *it performs fewer than* $m(\mathrm{D}(\mathcal{V})+(m+1)/2)$ *row reductions and has asymptotic complexity* $O(m^2 \mathrm{D}(\mathcal{V})N)$ *where* N *is the maximal degree of any term in* $\mathcal{V}$.

Proof If Algorithm 1 terminates, the output matrix must be unimodular equivalent to the input since it is reached by a finite number of row-operations. Since we can apply row reductions on a matrix if and only if it is not in weak Popov form, Algorithm 1 must bring $\mathcal{V}$ to this form.

Termination follows directly from Lemma 12 since the value of a row decreases each time a row reduction is performed. To be more precise, we furthermore see that the maximal number of row reductions performed on $\mathcal{V}$ before reaching a matrix $\mathcal{U}$ in weak Popov form is at most $\sum_{i=0}^{m-1} \psi(\mathbf{v}_i) - \psi(\mathbf{u}_i)$. Expanding this, we get

$$\begin{aligned}\sum_{i=0}^{m-1} \psi(\mathbf{v}_i) - \psi(\mathbf{u}_i) &= \sum_{i=0}^{m-1} \big(m(\deg \mathbf{v}_i - \deg \mathbf{u}_i) + \mathrm{LP}(\mathbf{v}_i) - \mathrm{LP}(\mathbf{u}_i)\big) \\ &= m(\deg \mathcal{V} - \deg \mathcal{U}) + \textstyle\sum_{i=0}^{m-1} \mathrm{LP}(\mathbf{v}_i) - \binom{m}{2} \\ &< m\Big(\mathrm{D}(\mathcal{V}) + \tfrac{m+1}{2}\Big)\end{aligned}$$

where we use $\deg \mathcal{U} = \deg \det \mathcal{U} = \deg \det \mathcal{V}$ and that the $\mathrm{LP}(\mathbf{u}_i)$ are all different.

For the asymptotic complexity, note that during the algorithm, no polynomial in the matrix will have larger degree than N. The estimate is reached simply by remarking that one row reduction consists of m times scaling and adding two such polynomials.

□

Let us consider an example to illustrate all the above.

Example 14 (Orthogonality Defect and Weak-Popov Form) *Let us consider the following matrices* $\mathcal{V}_0, \ldots, \mathcal{V}_3 \in \mathbb{F}_2[X]^{3\times 3}$. *From matrix* $\mathcal{V}_i$ *to* $\mathcal{V}_{i+1}$ *we performed one row-operation:*

$$\mathcal{V}_0 = \begin{pmatrix} 1 & X^2 & X \\ 0 & X^3 & X^2 \\ X & 1 & 0 \end{pmatrix} \xrightarrow{(0,1)} \mathcal{V}_1 = \begin{pmatrix} 1 & X^2 & X \\ X & 0 & 0 \\ X & 1 & 0 \end{pmatrix} \xrightarrow{(2,1)} \mathcal{V}_2 = \begin{pmatrix} 1 & X^2 & X \\ X & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \xrightarrow{(0,2)} \mathcal{V}_3 = \begin{pmatrix} 1 & 0 & X \\ X & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

where the indexes (i_1, i_2) *on the arrow indicated the concerned rows. The orthogonality defect is decreasing;* $\mathrm{D}(\mathcal{V}_0) = 3 \to \mathrm{D}(\mathcal{V}_1) = 2 \to \mathrm{D}(\mathcal{V}_2) = 1 \to \mathrm{D}(\mathcal{V}_3) = 0$, *and* $\mathcal{V}_3$ *is in weak Popov form.*

In [1], Alekhnovich gave a divide-&-conquer variant of the Mulders–Storjohann-algorithm: the same row reductions are performed but structured in a binary computation tree, where work is done on matrices of progressively smaller degree towards the bottom, ending with essentially $\mathbb{F}_q$-matrices at the leaves. Alekhnovich does not seem to have been aware of the work of Mulders and Storjohann, and basically reinvented their algorithm before giving his divide-&-conquer variant.

For square matrices, we can improve upon the complexity analysis that Alekhnovich gave by using the concept of orthogonality defect; this will be crucial for our aims.

Lemma 15 (Alekhnovich's Algorithm [1]) *Alekhnovich's algorithm inputs a matrix* $\mathcal{V} \in \mathbb{F}_q[X]^{m\times m}$ *and outputs a unimodular equivalent matrix which is in weak Popov form. Let* N *be the greatest degree of a term in* $\mathcal{V}$. *If* $N \in O(\mathrm{D}(\mathcal{V}))$ *then the algorithm has asymptotic complexity:*

$$O\big(m^3 \,\mathrm{D}(\mathcal{V}) \log^2 \mathrm{D}(\mathcal{V}) \log\log \mathrm{D}(\mathcal{V})\big) \quad \textit{operations over } \mathbb{F}_q.$$

Proof The description of the algorithm as well as the proof of its correctness can be found in [1]. We only prove the claim on the complexity. The method $R(\mathcal{V}, t)$ of [1] computes a unimodular matrix $\mathcal{U}$ such that $\deg(\mathcal{U}\mathcal{V}) \le \deg \mathcal{V} - t$ or $\mathcal{U}\mathcal{V}$ is in weak Popov form. According to [1, Lemma 2.10], the asymptotic complexity of this computation

is in $O(m^3 t \log^2 t \log\log t)$. Due to Lemma 10, we can set $t = \mathrm{D}(\mathcal{V})$ to be sure that $\mathcal{UV}$ is in weak Popov form. What remains is just to compute the product $\mathcal{UV}$. Due to [1, Lemma 2.8], each entry in $\mathcal{U}$ can be represented as $p(X)X^d$ for some $d \in \mathbb{N}_0$ and $p(X) \in \mathbb{F}_q[X]$ of degree at most $2t$. If therefore $N \in O(\mathrm{D}(\mathcal{V}))$, the complexity of performing the matrix multiplication using the naive algorithm is $O(m^3 \mathrm{D}(\mathcal{V}))$. □

The Beelen–Brander interpolation algorithm [3] works simply by computing $\mathcal{A}_{s,\ell}$ and then applying Alekhnovich's algorithm on $\mathcal{A}_{s,\ell}\mathcal{W}_\ell$; a minimal $(1, k-1)$-weighted polynomial in $M_{s,\ell}$ can then be directly retrieved as a row in the reduced matrix, after removing $\mathcal{W}_\ell$. The algorithm's complexity therefore follows from the above theorem once we have computed the orthogonality defect of $\mathcal{A}_{s,\ell}\mathcal{W}_\ell$:

Lemma 16 $\mathrm{D}(\mathcal{A}_{s,\ell}\mathcal{W}_\ell) = \frac{1}{2}(2\ell - s + 1)s(\deg R - k + 1) \;<\; \ell s(n-k)$.

Proof We will compute first $\deg(\mathcal{A}_{s,\ell}\mathcal{W}_\ell)$ and then $\deg\det(\mathcal{A}_{s,\ell}\mathcal{W}_\ell)$.

For the former, we have $\deg(\mathcal{A}_{s,\ell}\mathcal{W}_\ell) = \sum_{t=0}^{\ell} \mathrm{wdeg}_{1,k-1} P^{(t)}(X,Y)$, where the $P^{(t)}(X,Y)$ are as in Theorem 5. Note that whenever $\mathbf{r}$ is not a codeword then $\deg R \geq k$. Therefore, $\mathrm{wdeg}_{1,k-1}(Y - R(X))^t = t \deg R(X)$ and so

$$\begin{aligned}
\mathrm{wdeg}_{1,k-1} P^{(t)}(X,Y) &= (s-t)n + t\deg R(X) && \text{for } t < s,\\
\mathrm{wdeg}_{1,k-1} P^{(t)}(X,Y) &= (t-s)(k-1) + s\deg R(X) && \text{for } t \geq s.
\end{aligned}$$

This gives

$$\deg(\mathcal{A}_{s,\ell}\mathcal{W}_\ell) = \tbinom{s+1}{2} n + \tbinom{\ell-s+1}{2}(k-1) + \left(\tbinom{s+1}{2} + (\ell-s)s\right)\deg R(X).$$

Since $\mathcal{A}_{s,\ell}$ is lower triangular, the determinant is:

$$\begin{aligned}
\det(\mathcal{A}_{s,\ell}\mathcal{W}_\ell) &= \prod_{t=0}^{s} G^{s-t} \prod_{t=0}^{\ell} X^{t(k-1)}, && \text{and so}\\
\deg\det(\mathcal{A}_{s,\ell}\mathcal{W}_\ell) &= \tbinom{s+1}{2} n + \tbinom{\ell+1}{2}(k-1).
\end{aligned}$$

The orthogonality defect can then be simplified to

$$\begin{aligned}
\mathrm{D}(\mathcal{A}_{s,\ell}\mathcal{W}_\ell) &= \tbinom{\ell-s+1}{2}(k-1) + \left(\tbinom{s+1}{2} + (\ell-s)s\right)\deg R(X) - \tbinom{\ell+1}{2}(k-1)\\
&= \deg R(X)\left(s\ell - \tfrac{1}{2}s^2 + \tfrac{1}{2}s\right) - \tfrac{1}{2}(k-1)\left(\ell^2 + \ell - (\ell-s+1)(\ell-s)\right)\\
&= \tfrac{1}{2}(2\ell - s + 1)s(\deg R(X) - k + 1).
\end{aligned}$$

□

Remark In the earlier work of Lee and O'Sullivan [16], they construct basically the same matrix as $\mathcal{A}_{s,\ell}$, but apply their own Gröbner basis algorithm on this. They also does not seem to have been aware of the work of Mulders and Storjohann [18], but their algorithm is basically a variant of Algorithm 1 which keeps the rows in a specific order.

4 Multi-Trial List Decoding

4.1 Basic Idea

Using the results of the preceding section, we show in Section 4.3 that, given a basis of $M_{s,\ell}$ as a matrix $\mathcal{B}_{s,\ell}$ in weighted weak Popov form, then we can write down a matrix $\mathcal{C}^{\mathrm{I}}_{s,\ell+1}$ which is a basis of $M_{s,\ell+1}$ and where $\mathrm{D}(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_\ell)$ is much lower than $\mathrm{D}(\mathcal{A}_{s,\ell+1}\mathcal{W}_\ell)$. This means that reducing $\mathcal{C}^{\mathrm{I}}_{s,\ell+1}$ to weighted weak Popov form using Alekhnovich's algorithm [1] is faster than reducing $\mathcal{A}_{s,\ell+1}$. We call this kind of refinement a "micro-step of type I". In Section 4.4, we similarly give a way to refine a basis of $M_{s,\ell}$ to one of $M_{s+1,\ell+1}$, and we call this a "micro-step of type II".

If we first compute a basis in weighted weak Popov form of $M_{1,1}$ using $\mathcal{A}_{1,1}$, we can perform a sequence of micro-steps of type I and II to compute a basis in weighted weak Popov form of $M_{s,\ell}$ for any s, ℓ with $\ell \geq s$. After any step, having some intermediate $\hat{s} \leq s$, $\hat{\ell} \leq \ell$, we will thus have a basis of $M_{\hat{s},\hat{\ell}}$ in weighted weak Popov form. By Corollary 8, we could extract from $\mathcal{B}_{\hat{s},\hat{\ell}}$ a $\hat{Q}(X,Y) \in M_{\hat{s},\hat{\ell}}$ with minimal $(1, k-1)$-weighted degree. Since it must satisfy the interpolation conditions of Theorem 1, and since the weighted degree is minimal among such polynomials, it must also satisfy the degree constraints for $\hat{\tau} = \tau(\hat{s}, \hat{\ell})$. By that theorem any codeword with distance at most $\hat{\tau}$ from $\mathbf{r}$ would then be represented by a root of $\hat{Q}(X,Y)$.

Algorithm 2 is a generalisation and formalisation of this method. For a given $\mathcal{GRS}(n,k)$ code, one chooses ultimate parameters (s, ℓ, τ) being a permissible triple with $s \leq \ell$. One also chooses a list of micro-steps and chooses after which micro-steps to attempt decoding; these choices are represented by a list C consisting of S_1, S_2 and Root elements. This list must contain exactly $\ell - s$ S_1-elements and $s-1$ S_2-elements, as it begins by computing a basis for $M_{1,1}$ and will end with a basis for $M_{s,\ell}$. Whenever there is a Root element in the list, the algorithm performs root-finding and finds all codewords with distance at most $\hat{\tau} = \tau(\hat{s}, \hat{\ell})$ from $\mathbf{r}$; if this list is non-empty, the computation breaks and the list is returned.

The algorithm calls sub-functions which we explain informally: $\mathsf{MicroStep1}$ and $\mathsf{MicroStep2}$ will take $\hat{s}, \hat{\ell}$ and a basis in weighted weak Popov form for $M_{\hat{s},\hat{\ell}}$ and return a basis in weighted weak Popov form for $M_{\hat{s},\hat{\ell}+1}$ respectively $M_{\hat{s}+1,\hat{\ell}+1}$; more detailed descriptions for these are given in Subsections 4.3 and 4.4. $\mathsf{MinimalWeightedRow}$ finds a polynomial of minimal $(1, k-1)$-weighted degree in $M_{\hat{s},\hat{\ell}}$ given a basis in weighted weak Popov form (Corollary 8). Finally, $\mathsf{RootFinding}(Q, \tau)$ returns all Y-roots of $Q(X,Y)$ of degree less than k and whose corresponding codeword has distance at most τ from the received word $\mathbf{r}$.

The correctness of Algorithm 2 for any possible choice of s, ℓ and C follows from our discussion as well as Sections 4.3 and 4.4. Before going to these two technical sections, we will discuss what possibilities the micro-steps of type I and II offer, and in particular, do not, with regards to decoding radii.

In the following two subsections we explain the details of the micro-steps. In Section 4.5, we discuss the complexity of the method and how the choice of C influences this.

Algorithm 2: Multi-Trial Guruswami–Sudan Decoding

Input:
A $\mathcal{GRS}(n,k)$ code over $\mathbb{F}_q$ with $w_0,\ldots,w_{n-1} \in \mathbb{F}_q^*$
The received vector $\mathbf{r} = (r_0,\ldots,r_{n-1}) \in \mathbb{F}_q^n$
A permissible triple $(s,\ell,\tau) \in \mathbb{N}^3$
A list C with elements in $\{\mathsf{S}_1, \mathsf{S}_2, \mathsf{Root}\}$ with $\ell - s$ instances of S_1 and $s-1$ instances of S_2

Preprocessing:
Calculate $r'_i = r_i/w_i$ for all $i = 0,\ldots,n-1$
Construct $\mathcal{A}_{1,1}$, and compute $\mathcal{B}_{1,1}$ from $\mathcal{A}_{1,1}\mathcal{W}_1$ using Alekhnovich's algorithm
Initial parameters $(\hat{s},\hat{\ell}) \leftarrow (1,1)$

```
for each c in C do
    if c = S_1 then
        B_{ŝ,ℓ̂+1}  ← MicroStep1(ŝ, ℓ̂, B_{ŝ,ℓ̂})
        (ŝ, ℓ̂)     ← (ŝ, ℓ̂ + 1)
    if c = S_2 then
        B_{ŝ+1,ℓ̂+1} ← MicroStep2(ŝ, ℓ̂, B_{ŝ,ℓ̂})
        (ŝ, ℓ̂)     ← (ŝ + 1, ℓ̂ + 1)
    if c = Root then
        Q(X,Y) ← MinimalWeightedRow(B_{ŝ,ℓ̂})
        if RootFinding(Q(X,Y), τ(ŝ, ℓ̂)) ≠ ∅ then
            return this list
```

4.2 The Possible Refinement Paths

The choice of C provides much flexibility to the algorithm. The two extreme cases are perhaps the most generally interesting: the one without any Root elements except at the end, i.e., usual list-decoding; and the one with a Root element each time the intermediate decoding radius $\hat{\tau}$ has increased, i.e., a variant of maximum-likelihood decoding up to a certain radius.

In Section 4.5, we discuss complexity concerns with regards to the chosen path; it turns out that the price of either type of micro-step is very comparable, and the worst-case complexity is completely unchanged by the choice of C. However, in the case where we have multiple Root elements we want to minimise the *average* computation cost: considering that few errors occur much more frequently than many, we should therefore seek to reach each intermediate decoding radius after as few micro-steps as possible.

Since we do not have a refinement which increases only s, we are inherently limited in the possible paths we can choose, so the question arises if this limitation conflicts with our interest as given above.

First — and most important — for any given final decoding radius τ, we mentioned in Section 2.2 that the corresponding parameters satisfy $s < \ell$, and so we can reach these values using only micro-steps of type I and II.

For the intermediate steps, the strongest condition we would like to have satisfied is the following: Let $d/2 \le \tau_1 < \ldots < \tau_m = \tau$ be the series of intermediate decoding radii where we would like to attempt decoding. Let (s_i,ℓ_i) be chosen such that (s_i,ℓ_i,τ_i)

is permissible and either s_i or ℓ_i is minimal possible for the given τ_i. Can then the sequence of parameters (s_i, ℓ_i) be reached by a series of micro-steps of type I and II?

Unfortunately, we do not have a formal proof of this statement. However, we have verified for a large number of parameters that it is true.

4.3 Micro-Step Type I: $(s, \ell) \mapsto (s, \ell + 1)$

The function MicroStep1 is based on the following lemma:

Lemma 17 *If $B^{(0)}(X,Y), \ldots, B^{(\ell)}(X,Y) \in \mathbb{F}_q[X,Y]$ is a basis of $M_{s,\ell}$, then the following is a basis of $M_{s,\ell+1}$:*

$$B^{(0)}(X,Y),\ \ldots\ ,\ B^{(\ell)}(X,Y), Y^{\ell-s+1}(Y - R(X))^s.$$

Proof In the basis of $M_{s,\ell+1}$ given in Theorem 5, the first $\ell + 1$ generators are the generators of $M_{s,\ell}$. Thus, all of these can be described by any basis of $M_{s,\ell+1}$. The last remaining generator is exactly $Y^{\ell-s+1}(Y - R(X))^s$. □

In particular, the above lemma holds for a basis of $M_{s,\ell+1}$ in weighted weak Popov form, represented by a matrix $\mathcal{B}_{s,\ell}$. The following matrix thus represents a basis of $M_{s,\ell+1}$:

$$\mathcal{C}^{\mathrm{I}}_{s,\ell+1} = \left(\begin{array}{c|c} \mathcal{B}_{s,\ell} & \mathbf{0}^T \\ \hline 0 \ldots 0\ (-R)^s\ \binom{s}{1}(-R)^{s-1}\ \ldots\ 1 & \end{array}\right). \tag{6}$$

Lemma 18 $\mathrm{D}(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_{\ell+1}) = s(\deg R - k + 1) \le s(n - k)$.

Proof We calculate the two quantities $\det(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_{\ell+1})$ and $\deg(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_{\ell+1})$. It is easy to see that

$$\det(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_{\ell+1}) = \det \mathcal{B}_{s,\ell} \det \mathcal{W}_{\ell+1} = \det \mathcal{B}_{s,\ell} \det \mathcal{W}_\ell X^{(\ell+1)(k-1)}.$$

For the row-degree, this is clearly $\deg(\mathcal{B}_{s,\ell}\mathcal{W}_\ell)$ plus the row-degree of the last row. If and only if the received word is not a codeword then $\deg R \ge k$, so the leading term of the last row must be $(-R)^s X^{(\ell+1-s)(k-1)}$. Thus, we get

$$\begin{aligned} \mathrm{D}(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_{\ell+1}) &= \big(\deg(\mathcal{B}_{s,\ell}\mathcal{W}_\ell) + s \deg R + (\ell + 1 - s)(k-1)\big) \\ &\quad - \big(\deg \det(\mathcal{B}_{s,\ell}\mathcal{W}_\ell) + (\ell+1)(k-1)\big) \\ &= s(\deg R - k + 1), \end{aligned}$$

where the last step follows from Lemma 10 as $\mathcal{B}_{s,\ell}\mathcal{W}_\ell$ is in weak Popov form. □

Corollary 19 *The complexity of* MicroStep1$(s, \ell, \mathcal{B}_{s,\ell})$ *is* $O(\ell^3 sn \log^2 n \log\log n)$.

Proof Follows by Lemma 15. Since $s \in O(n^2)$ we can leave out the s in log-terms. □

4.4 Micro-Step Type II: $(s, \ell) \mapsto (s+1, \ell+1)$

The function MicroStep2 is based on the following lemma:

Lemma 20 *If $B^{(0)}(X,Y), \ldots, B^{(\ell)}(X,Y) \in \mathbb{F}_q[X,Y]$ is a basis of $M_{s,\ell}$, then the following is a basis of $M_{s+1,\ell+1}$:*

$$G^{s+1}(X),\ B^{(0)}(X,Y)(Y - R(X)),\ \ldots\ ,\ B^{(\ell)}(X,Y)(Y - R(X)).$$

Proof Denote by $P^{(0)}_{s,\ell}(X,Y), \ldots, P^{(\ell)}_{s,\ell}(X,Y)$ the basis of $M_{s,\ell}$ as given in Theorem 5, and by $P^{(0)}_{s+1,\ell+1}(X,Y), \ldots, P^{(\ell+1)}_{s+1,\ell+1}(X,Y)$ the basis of $M_{s+1,\ell+1}$. Then observe that for $t > 0$, we have $P^{(t)}_{s+1,\ell+1} = P^{(t-1)}_{s,\ell}(Y - R(X))$. Since the $B^{(t)}(X,Y)$ form a basis of $M_{s,\ell}$, each $P^{(t)}_{s,\ell}$ is expressible as an $\mathbb{F}_q[X]$-combination of these, and thus for $t > 0$, $P^{(t)}_{s+1,\ell+1}$ is expressible as an $\mathbb{F}_q[X]$-combination of the $B^{(t)}(X,Y)(Y - R(X))$. Remaining is then only $P^{(0)}_{s+1,\ell+1}(X,Y) = G^{s+1}(X)$. □

As before, we can use the above with the basis $\mathcal{B}_{s,\ell}$ of $M_{s,\ell}$ in weighted weak Popov form, found in the previous iteration of our algorithm. Recall that multiplying by Y translates in the matrix representation to shifting one column to the right, so the following matrix represents a basis of $M_{s+1,\ell+1}$:

$$\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1} = \left(\begin{array}{c|c} G^{s+1} & \mathbf{0} \\ \hline \mathbf{0}^T & \mathbf{0} \end{array}\right) + \left(\begin{array}{c|c} 0 & \mathbf{0} \\ \hline \mathbf{0}^T & \mathcal{B}_{s,\ell} \end{array}\right) - R \cdot \left(\begin{array}{c|c} \mathbf{0} & 0 \\ \hline \mathcal{B}_{s,\ell} & \mathbf{0}^T \end{array}\right). \tag{7}$$

Lemma 21 $\mathrm{D}(\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1}) = (\ell+1)(\deg R - k + 1) \le (\ell+1)(n-k)$.

Proof We compute $\deg(\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1})$ and $\deg\det(\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1})$. For the former, obviously the first row has degree $(s+1)n$. Let $\mathbf{b}_i$ denote the ith row of $\mathcal{B}_{s,\ell}$ and $\mathbf{b}'_i$ denote the ith row of $\mathcal{B}_{s,\ell}\mathcal{W}_\ell$. The $(i+1)$th row of $\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1}$ has the form

$$\big[(0 \mid \mathbf{b}_i) - R(\mathbf{b}_i \mid 0)\big]\mathcal{W}_{\ell+1} = (0 \mid \mathbf{b}'_i)X^{k-1} - R(\mathbf{b}'_i \mid 0).$$

If and only if the received word is not a codeword, then $\deg R \ge k$. In this case, the leading term of $R\mathbf{b}'_i$ must have greater degree than any term in $X^{k-1}\mathbf{b}'_i$. Thus the degree of the above row is $\deg R + \deg \mathbf{b}'_i$. Summing up we get

$$\begin{aligned}\deg \mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1} &= (s+1)n + \sum_{i=0}^{\ell}\big(\deg R + \deg \mathbf{b}'_i\big) \\ &= (s+1)n + (\ell+1)\deg R + \deg(\mathcal{B}_{s,\ell}\mathcal{W}_\ell).\end{aligned}$$

For the determinant, observe that

$$\begin{aligned}\det(\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1}) &= \det(\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1})\det(\mathcal{W}_{\ell+1}) \\ &= G^{s+1}\det\widetilde{\mathcal{B}}\det\mathcal{W}_\ell \cdot X^{(\ell+1)(k-1)},\end{aligned}$$

where $\widetilde{\mathcal{B}} = \mathcal{B}_{s,\ell} - R\left(\grave{\mathcal{B}}_{s,\ell} \mid \mathbf{0}^T\right)$ and $\grave{\mathcal{B}}_{s,\ell}$ is all but the zeroth column of $\mathcal{B}_{s,\ell}$. This means $\widetilde{\mathcal{B}}$ can be obtained by starting from $\mathcal{B}_{s,\ell}$ and iteratively adding the $(j+1)$th column of $\mathcal{B}_{s,\ell}$ scaled by $R(X)$ to the jth column, with j starting from 0 up to $\ell - 1$. Since each of these will add a scaled version of an existing column in the matrix, this does not change the determinant. Thus, $\det \widetilde{\mathcal{B}} = \det \mathcal{B}_{s,\ell}$. But then $\det \widetilde{\mathcal{B}} \det \mathcal{W}_\ell = \det(\mathcal{B}_{s,\ell}\mathcal{W}_\ell)$ and so $\deg(\det \widetilde{\mathcal{B}} \det \mathcal{W}_\ell) = \deg(\mathcal{B}_{s,\ell}\mathcal{W}_\ell)$ by Lemma 10 since $\mathcal{B}_{s,\ell}\mathcal{W}_\ell$ is in weak Popov form. Thus we get

$$\deg \det(\mathcal{C}^{\mathrm{II}}_{s+1,\ell+1}\mathcal{W}_{\ell+1}) = (s+1)n + \deg(\mathcal{B}_{s,\ell}\mathcal{W}_\ell) + (\ell+1)(k-1).$$

The lemma follows from the difference of the two calculated quantities. □

Corollary 22 *The complexity of* $\mathsf{MicroStep2}(s, \ell, \mathcal{B}_{s,\ell})$ *is* $O(\ell^4 n \log^2 n \log\log n)$.

Example 23 *We consider again the* $\mathcal{GRS}(16, 4)$ *code over* $\mathbb{F}_{17}$ *of Example 3, and specify now that* $\alpha_i = i + 1$ *and* $w_i = 1$ *for* $i = 0, \ldots, 15$. *The aimed decoding radius is* $\tau = 8$ *and therefore the permissible triple* $(s, \ell, \tau) = (2, 4, 8)$ *should be reached iteratively by Algorithm 2. To maximise the decoding radius during the procedure, we could choose the following sequence of intermediate parameters* $(\hat{s}, \hat{\ell}, \hat{\tau})$*:*

$$(1, 1, 6) \xrightarrow{\mathrm{I}} (1, 2, 7) \xrightarrow{\mathrm{II}} (2, 3, 7) \xrightarrow{\mathrm{I}} (2, 4, 8).$$

We perform root-finding only if the decoding radius is increased. Therefore, the list C *of operations becomes:*

$$\mathsf{C} = \left\{\mathsf{Root}, \mathsf{S}_1, \mathsf{Root}, \mathsf{S}_2, \mathsf{S}_1, \mathsf{Root}\right\}.$$

With the information polynomial $f(X) = 2X^2 + 10X + 6$, *we obtain with* (2) *the following codeword:*

$$\mathbf{c} = (1, 0, 3, 10, 4, 2, 4, 10, 3, 0, 1, 6, 15, 11, 11, 15).$$

Consider that $\mathbf{r} = (1, 15, 12, 13, 4, 7, 4, 10, 1, 0, 1, 10, 2, 11, 11, 10)$ *was received, i.e., that the error* $\mathbf{e} = (0, 15, 9, 3, 0, 5, 0, 0, 15, 0, 0, 4, 4, 0, 0, 12)$ *of weight 8 occurred.*

We will depict the degrees of the polynomials in the matrices in the iterative decoding process. These are for this particular received word, but for a generic received word, the degrees are the same. For some $p(X) \in \mathbb{F}_q[X]$, *we will write* $p(X) \preceq t$ *for* $t \in \mathbb{N}_0$ *if* $\deg p(X) = t$, *and* $p(X) \preceq \perp$ *if* $p(X) = 0$, *and we extend* $\preceq$ *element-wise to matrices. To begin with, we have:*

$$\mathcal{A}_{1,1} = \begin{pmatrix} G & 0 \\ -R & 1 \end{pmatrix} \quad \mathcal{A}_{1,1} \preceq \begin{pmatrix} 16 & \perp \\ 15 & 0 \end{pmatrix} \quad \mathcal{A}_{1,1}\mathcal{W}_1 \preceq \begin{pmatrix} 16 & \perp \\ 15 & 3 \end{pmatrix}$$

according to (4) *and* (5). *We then apply Alekhnovich's algorithm on* $\mathcal{A}_{1,1}\mathcal{W}_1$ *to obtain* $\mathcal{B}_{1,1}\mathcal{W}_1$ *which is in weak Popov form. From this we can easily scale down the columns again to obtain* $\mathcal{B}_{1,1}$. *It took 11 row reductions, while* $(\hat{\ell}+1)(\mathrm{D}(\mathcal{A}_{1,1}\mathcal{W}_1) + \hat{\ell} + 1) = 28$ *was the upper bound, according to Lemma 13. We obtain*

$$\mathcal{B}_{1,1} \preceq \begin{pmatrix} 10 & 6 \\ 9 & 6 \end{pmatrix} \qquad \mathcal{B}_{1,1}\mathcal{W}_1 \preceq \begin{pmatrix} 10 & 9 \\ 9 & 9 \end{pmatrix}.$$

The first element in C *is* Root*, so we pick the second row of* $\mathcal{B}_{1,1}$*, since it has weighted degree less than* 10*, and we interpret it as a polynomial:*

$$\begin{aligned} Q_{1,1}(X,Y) = (14X^6 + 9X^4 + 9X^3 + 14X^2 + 4X + 1)Y \\ + 13X^9 + 10X^8 + 7X^6 + 16X^5 + 8X^3 + 12X^2 + 3X + 16. \end{aligned}$$

Root-finding of $Q_{1,1}(X,Y)$ *yields no results. The next element in* C *is* S_1*, so we move to the next intermediate parameters* $(\hat{s}, \hat{\ell}, \hat{\tau}) = (1, 2, 7)$*. From* (6)*, we get*

$$\mathcal{C}^{\mathrm{I}}_{1,2} = \left(\begin{array}{c|c} \mathcal{B}_{1,1} & \begin{matrix}0\\0\end{matrix} \\ \hline \begin{matrix}0 & -R\end{matrix} & 1 \end{array}\right) \qquad \textit{and so} \qquad \mathcal{C}^{\mathrm{I}}_{1,2} \preceq \begin{pmatrix} 10 & 6 & \bot \\ 9 & 6 & \bot \\ \bot & 15 & 0 \end{pmatrix} \qquad \mathcal{C}^{\mathrm{I}}_{1,2}\mathcal{W}_2 \preceq \begin{pmatrix} 10 & 9 & \bot \\ 9 & 9 & \bot \\ \bot & 18 & 6 \end{pmatrix}.$$

Running Alekhnovich's algorithm on $\mathcal{C}^{\mathrm{I}}_{1,2}\mathcal{W}_2$*, we obtain:*

$$\mathcal{B}_{1,2} \preceq \begin{pmatrix} 9 & 4 & 2 \\ 8 & 5 & 2 \\ 8 & 5 & 1 \end{pmatrix} \qquad \mathcal{B}_{1,2}\mathcal{W}_2 \preceq \begin{pmatrix} 9 & 7 & 8 \\ 8 & 8 & 8 \\ 8 & 8 & 7 \end{pmatrix}.$$

Since $\mathrm{D}(\mathcal{B}_{1,2}\mathcal{W}_2) = 12$*, Lemma 13 gives* 45 *as the upper bound on the number of row reductions, but it was done with only 24.*

In the next iteration we again meet a Root*. For our polynomial we can pick either the second or third row of* $\mathcal{B}_{1,2}$ *since both have weighted degree* $8 < \hat{s}(n - \hat{\tau}) = 9$*; we choose the second and obtain:*

$$\begin{aligned} Q_{1,2}(X,Y) = (15X^2 + 8X)Y^2 + (5X^5 + 2X^4 + 2X^3 + 10X^2 + 5X + 1)Y \\ + 14X^8 + 16X^7 + 7X^6 + 8X^5 + 9X^4 + 9X^3 + X^2 + 9X + 15. \end{aligned}$$

Again root-finding yields no results. The next element in C *is* S_2 *and we get intermediate parameters* $(\hat{s}, \hat{\ell}, \hat{\tau}) = (2, 3, 7)$*. We construct* $\mathcal{C}^{\mathrm{II}}_{2,3}$ *according to* (7) *which gives:*

$$\mathcal{C}^{\mathrm{II}}_{2,3} \preceq \begin{pmatrix} 32 & \bot & \bot & \bot \\ 24 & 19 & 17 & 2 \\ 23 & 20 & 17 & 2 \\ 23 & 20 & 16 & 1 \end{pmatrix} \qquad \mathcal{C}^{\mathrm{II}}_{2,3}\mathcal{W}_3 \preceq \begin{pmatrix} 32 & \bot & \bot & \bot \\ 24 & 22 & 23 & 11 \\ 23 & 23 & 23 & 11 \\ 23 & 23 & 22 & 10 \end{pmatrix}.$$

We needed 90 row reductions to reduce $\mathcal{C}^{\mathrm{II}}_{2,3}\mathcal{W}_3$ *to weak Popov form, while the upper bound is 160, since we calculated* $\mathrm{D}(\mathcal{C}^{\mathrm{II}}_{2,3}\mathcal{W}_3) = 36$*. After row-reduction, we obtain* $\mathcal{B}_{2,3}\mathcal{W}_3$*:*

$$\mathcal{B}_{2,3} \preceq \begin{pmatrix} 17 & 14 & 10 & 6 \\ 17 & 13 & 10 & 7 \\ 16 & 13 & 9 & 7 \\ 16 & 13 & 10 & 6 \end{pmatrix} \qquad \mathcal{B}_{2,3}\mathcal{W}_3 \preceq \begin{pmatrix} 17 & 17 & 16 & 15 \\ 17 & 16 & 16 & 16 \\ 16 & 16 & 15 & 16 \\ 16 & 16 & 16 & 15 \end{pmatrix}.$$

The next element in C *is* S_2*, so we construct* $\mathcal{C}^{\mathrm{II}}_{2,4}$ *according to* (7) *and get:*

$$\mathcal{C}^{\mathrm{II}}_{2,4} \preceq \begin{pmatrix} 17 & 14 & 10 & 6 & \bot \\ 17 & 13 & 10 & 7 & \bot \\ 16 & 13 & 9 & 7 & \bot \\ 16 & 13 & 10 & 6 & \bot \\ \bot & \bot & 30 & 15 & 0 \end{pmatrix} \qquad \mathcal{C}^{\mathrm{II}}_{2,4}\mathcal{W}_4 \preceq \begin{pmatrix} 17 & 17 & 16 & 15 & \bot \\ 17 & 16 & 16 & 16 & \bot \\ 16 & 16 & 15 & 16 & \bot \\ 16 & 16 & 16 & 15 & \bot \\ \bot & \bot & 36 & 24 & 12 \end{pmatrix}$$

$$\mathcal{B}_{2,4} \preceq \begin{pmatrix} 16 & 13 & 10 & 6 & 3 \\ 15 & 13 & 8 & 6 & 3 \\ 16 & 12 & 9 & 6 & 3 \\ 14 & 12 & 9 & 6 & 3 \\ 14 & 12 & 9 & 6 & 2 \end{pmatrix} \qquad \mathcal{B}_{2,4}\mathcal{W}_4 \preceq \begin{pmatrix} 16 & 16 & 16 & 15 & 15 \\ 15 & 16 & 14 & 15 & 15 \\ 16 & 15 & 15 & 15 & 15 \\ 14 & 15 & 15 & 15 & 15 \\ 14 & 15 & 15 & 15 & 14 \end{pmatrix}.$$

We needed 86 row reductions for the module minimisation, while the upper bound was 145 since we calculated $\mathrm{D}(\mathcal{C}^{\mathrm{II}}_{2,4}\mathcal{W}_4) = 24$*.*

The last iteration is again a Root*, and we can use either of the two last rows of* $\mathcal{C}^{\mathrm{II}}_{2,4}$ *since they have weighted degree* $< s(n-\tau) = 16$*. Using the last, the obtained polynomial is:*

$$\begin{aligned} Q_{2,4}(X,Y) = {} & (6X^3 + 16X^2 + 10X)Y^4 + (11X^6 + 16X^5 + 14X^4 + 15X^2 + 5X + 1)Y^3 \\ & + (15X^9 + X^8 + 5X^7 + 6X^6 + 12X^5 + 2X^4 + 6X^3 + 2X^2 + 12X + 2)Y^2 \\ & + (5X^{12} + 5X^{11} + 2X^{10} + 9X^9 + 14X^8 + 6X^7 + 4X^6 + 3X^5 + 16X^4 \\ & + X^3 + 16X^2 + 6X)Y + 7X^{14} + 16X^{13} + 6X^{12} + 4X^{11} + 11X^{10} \\ & + 11X^8 + 4X^7 + 5X^6 + 16X^5 + 12X^4 + 15X^3 + 6X^2 + 16X + 1. \end{aligned}$$

Indeed, $Q_{2,4}(X, 2X^2 + 10X + 6) = 0$ *and root-finding retrieves* $f(X)$ *for us.*

As the example shows, performing module minimisation on a matrix can be informally seen to "balance" the row-degrees such that they all become roughly the same size. The complexity of this reduction depends on the number of row reductions, which in turn depends on the "unbalancedness" of the initial matrix. The matrices $\mathcal{C}^{\mathrm{I}}_{1,2}, \mathcal{C}^{\mathrm{II}}_{2,3}$ *and* $\mathcal{C}^{\mathrm{II}}_{2,4}$ *are more balanced in row-degrees than using* $\mathcal{A}_{1,2}, \mathcal{A}_{2,3}$ *and* $\mathcal{A}_{2,4}$ *directly.*

4.5 Complexity Analysis

Using the estimates of the two preceding subsections, we can make a rather precise worst-case asymptotic complexity analysis of Algorithm 2. The average running time will depend on the exact choice of C but we will see that the worst-case complexity will not. First, it is necessary to know the complexity of performing a root-finding attempt.

Lemma 24 (Complexity of Root-Finding) *Given a polynomial* $Q(X,Y) \in \mathbb{F}_q[X][Y]$ *of* Y*-degree at most* ℓ *and* X*-degree at most* N*, there exists an algorithm to find all* $\mathbb{F}_q[X]$*-roots of complexity* $O\big(\ell^2 N \log^2 N \log\log N\big)$*, assuming* $\ell, q \in O(N)$*.*

Proof We employ the Roth–Ruckenstein [21] root-finding algorithm together with the divide-and-conquer speed-up by Alekhnovich [1]. The complexity analysis in [1] needs to be slightly improved to yield the above, but see [4] for easy amendments. □

Theorem 25 (Complexity of Algorithm 2) *For a given $\mathcal{GRS}(n,k)$ code, as well as a given list of steps* C *for Algorithm 2 with ultimate parameters (s,ℓ,τ), the algorithm has worst-case complexity $O(\ell^4 sn\log^2 n\log\log n)$, assuming $q\in O(n)$.*

Proof The worst-case complexity corresponds to the case that we do not break early but run through the entire list C. Precomputing $\mathcal{A}_{1,1}$ using Lagrangian interpolation can be performed in $O(n\log^2 n\log\log n)$, see e.g. [9, p. 235], and reducing to $\mathcal{B}_{1,1}$ is in the same complexity by Lemma 15.

Now, C must contain exactly $\ell-s$ S_1-elements and $s-1$ S_2-elements. The complexities given in Corollaries 19 and 22 for some intermediate $\hat{s},\hat{\ell}$ can be relaxed to s and ℓ. Performing $O(\ell)$ micro-steps of type I and $O(s)$ of type II is therefore in $O(\ell^4 sn\log^2 n\log\log n)$.

It only remains to count the root-finding steps. Obviously, it never makes sense to have two Root after each other in C, so after removing such possible duplicates, there can be at most ℓ elements Root. When we perform root-finding for intermediate $\hat{s},\hat{\ell}$, we do so on a polynomial in $M_{\hat{s},\hat{\ell}}$ of minimal weighted degree, and by the definition of $M_{\hat{s},\hat{\ell}}$ as well as Theorem 1, this weighted degree will be less than $\hat{s}(n-\hat{\tau})<sn$. Thus we can apply Lemma 24 with $N=sn$. □

The worst-case complexity of our algorithm is equal to the average-case (and worst-case) complexity of the Beelen–Brander [3] list decoder. However, Theorem 25 shows that we can choose as many intermediate decoding attempts as we would like without changing the worst-case complexity. One could therefore choose to perform a decoding attempt just after computing $\mathcal{B}_{1,1}$ as well as every time the decoding radius has increased. The result would be a decoding algorithm finding all *closest* codewords within some ultimate radius τ. If one is working in a decoding model where such a list suffices, our algorithm will thus have much better average-case complexity since fewer errors occur more frequently than many.

5 Re-Encoding Transformation

We now discuss how to adjust Algorithm 2 to incorporate the re-encoding transformation proposed in [13,14]. The basic observation is that we can correct $\mathbf{r}$ if we can correct $\mathbf{r}-\hat{\mathbf{c}}$ for any $\hat{\mathbf{c}}\in\mathcal{GRS}(n,k)$. If we chose $\hat{\mathbf{c}}$ such that $\mathbf{r}-\hat{\mathbf{c}}$ for some reason is easier to handle in our decoder, we can save computational work. As in the original articles, we will choose $\hat{\mathbf{c}}$ such that it coincides with $\mathbf{r}$ in the first k positions; this can be done since it is just finding a Lagrange polynomial of degree $k-1$ that goes through these points. The re-encoded received word will therefore have 0 on the first k positions.

For ease of notation, assume that $\mathbf{r}$ is this re-encoded received word with first k positions zero, and we can reuse all the objects introduced in the preceding sections. Define

$$L(X)\triangleq\prod_{i=0}^{k-1}(X-\alpha_i). \tag{8}$$

Obviously $L(X)\mid G(X)$ so introduce $\bar{G}(X)=G(X)/L(X)$. However, since $r_i=0$ for $i<k$ then also $L(X)\mid R(X)$; this will be the observation which will save us computations. Introduce therefore $\bar{R}(X)=R(X)/L(X)$. Regard now $\mathcal{A}_{s,\ell}$ of (4); it is

clear that $L(X)^{s-t}$ divides every entry in the tth column for $t < s$. This implies that the image of the following bijective map is indeed $\mathbb{F}_q[X,Y]$:

$$\begin{aligned} \varphi: \quad M_{s,\ell} &\to \mathbb{F}_q[X,Y] \\ Q(X,Y) &\mapsto L(X)^{-s} Q(X, L(X)Y). \end{aligned} \tag{9}$$

Extend φ element-wise to sets of $M_{s,\ell}$ elements, and note that φ is therefore an isomorphism between $M_{s,\ell}$ and $\varphi(M_{s,\ell})$. The idea is now that the elements in $\varphi(M_{s,\ell})$ have lower X-degree than those in $M_{s,\ell}$, and we can therefore expect that working with bases of $\varphi(M_{s,\ell})$ is computationally cheaper than with bases of $M_{s,\ell}$. Since we are searching a minimal $(1, k-1)$-weighted polynomial in $M_{s,\ell}$, we need to be sure that this property corresponds to something sensible in $\varphi(M_{s,\ell})$. The following lemma and its corollary provides this:

Lemma 26 *For any $Q(X,Y)$ in $M_{s,\ell}$*

$$\operatorname{wdeg}_{1,k-1} Q(X,Y) = \operatorname{wdeg}_{1,-1} \varphi(Q(X,Y)) + sk.$$

Proof We have $\operatorname{wdeg}_{1,k-1} Q(X,Y) = \max_i \left\{ \deg Q_{[i]}(X) + i(k-1) \right\}$ so we obtain:

$$\begin{aligned} \operatorname{wdeg}_{1,-1} \varphi(Q(X,Y)) &= \max_i \left\{ \deg Q_{[i]}(X) - s \deg L(X) + i \deg L(X) - i \right\} \\ &= \max_i \left\{ \deg Q_{[i]}(X) + i(k-1) \right\} - sk. \end{aligned}$$

□

Corollary 27 *$Q(X,Y)$ has minimal $(1, k-1)$-weighted degree in $M_{s,\ell}$ if and only if $\varphi(Q(X,Y))$ has minimal $(1,-1)$-weighted degree in $\varphi(M_{s,\ell})$.*

Let us now describe the basis of $\varphi(M_{s,\ell})$ corresponding to the one in (4):

Theorem 28 *The module $\varphi(M_{s,\ell})$ is generated as an $\mathbb{F}_q[X]$-module by the $\ell+1$ polynomials $\bar{P}^{(t)}(X,Y) \in \mathbb{F}_q[X,Y]$ given by*

$$\begin{aligned} \bar{P}^{(t)}(X,Y) &= \bar{G}(X)^{s-t}(Y - \bar{R}(X))^t, && \text{for } 0 \le t < s, \\ \bar{P}^{(t)}(X,Y) &= (L(X)Y)^{t-s}(Y - \bar{R}(X))^s, && \text{for } s \le t \le \ell. \end{aligned}$$

Proof Follows directly from Theorem 5 and the mapping as defined (9).

We can represent the basis of $\varphi(M_{s,\ell})$ by the $(\ell+1) \times (\ell+1)$ matrix over $\mathbb{F}_q[X]$ (compare to (4)):

$$\bar{\mathcal{A}}_{s,\ell} \triangleq \begin{pmatrix} \bar{G}^s & & & & & & & \\ \bar{G}^{s-1}(-\bar{R}) & \bar{G}^{s-1} & & & & & & \\ \bar{G}^{s-2}(-\bar{R})^2 & 2\bar{G}^{s-2}(-\bar{R}) & \bar{G}^{s-2} & & & & 0 & \\ \vdots & & \ddots & & & & & \\ (-\bar{R})^s & \binom{s}{1}(-\bar{R})^{s-1} & \dots & 1 & & & & \\ & L(-\bar{R})^s & & \dots & L & & & \\ & & L^2(-\bar{R})^s & & \dots & L^2 & & \\ & & & \ddots & & & \ddots & \\ 0 & & & L^{\ell-s}(-\bar{R})^s & \dots & & L^{\ell-s} \end{pmatrix}. \tag{10}$$

We need an analogue of Corollary 8 for the $(1,-1)$-weighted degree, i.e., we should find a diagonal matrix to multiply on $\bar{\mathcal{A}}_{s,\ell}$ such that when module minimising the result, we will have a row corresponding to a polynomial in $\varphi(M_{s,\ell})$ with minimal $(1,-1)$-weighted degree. We cannot use $\operatorname{diag}(1, X^{-1}, \ldots, X^{-\ell})$, since multiplying with negative powers of X might cause us to leave the polynomial ring; however, we can to this add the same power to all the diagonal elements such that they become non-negative:

$$\bar{\mathcal{W}}_\ell = \operatorname{diag}\Big(X^\ell, X^{\ell-1}, \ldots, 1\Big). \tag{11}$$

Therefore, for a vector $\mathbf{q} = (Q_0(X)X^\ell, \ldots, Q_\ell(X)X^0)$ in the row-space of $\bar{\mathcal{A}}_{s,\ell}\bar{\mathcal{W}}_\ell$ corresponds a polynomial $Q(X,Y) = \sum_{t=0}^{\ell} Q_t(X)Y^t$, and we will have the identity $\deg \mathbf{q} = \operatorname{wdeg}_{1,-1} Q + \ell$. Obviously then, a minimal degree vector in $\bar{\mathcal{A}}_{s,\ell}\bar{\mathcal{W}}_\ell$ is a minimal $(1,-1)$-weighted polynomial in $\varphi(M_{s,\ell})$.

Finally, we need adjusted variants of the micro-steps I and II. The necessary adaptions of Algorithm 2 are summarised in the following lemma:

Lemma 29 *Let $\bar{\mathcal{B}} \in \mathbb{F}_q[X]^{(\ell+1)\times(\ell+1)}$ be the matrix representation of a basis of $\varphi(M_{s,\ell})$. If $\bar{\mathcal{B}}\bar{\mathcal{W}}_\ell$ is in weak Popov form, then one of the rows of $\bar{\mathcal{B}}$ corresponds to a polynomial in $\varphi(M_{s,\ell})$ with minimal $(1,-1)$-weighted degree.*

Modified micro-steps of type I and II can be obtained from the following. Let $\bar{B}^{(0)}(X,Y), \ldots, \bar{B}^{(\ell)}(X,Y) \in \mathbb{F}_q[X,Y]$ be a basis of $M_{s,\ell}$. Then the following is a basis of $M_{s,\ell+1}$:

$$\bar{B}^{(0)}(X,Y),\ \ldots\ ,\ \bar{B}^{(\ell)}(X,Y), (L(X)Y)^{\ell-s+1}(Y-\bar{R}(X))^s. \tag{12}$$

Similarly, the following is a basis of $M_{s+1,\ell+1}$:

$$\bar{G}^{s+1}(X),\ \bar{B}^{(0)}(X,Y)(Y-\bar{R}(X)),\ \ldots\ ,\ \bar{B}^{(\ell)}(X,Y)(Y-\bar{R}(X)). \tag{13}$$

Proof The first part follows from the previous discussion and analogously to Corollary 8. The recursive bases of (12) and (13) follow completely analogous to Lemmas 17 and 20, given Theorem 28. □

Example 30 *In the case of the $\mathcal{GRS}(16,4)$ code over $\mathbb{F}_{17}$ with final decoding radius $\tau = 8$ as shown in Example 23, then $\deg L(X) = 4$ and the initial matrix satisfies:*

$$\bar{\mathcal{A}}_{1,1} = \begin{pmatrix} \bar{G} & 0 \\ -\bar{R} & 1 \end{pmatrix} \quad \bar{\mathcal{A}}_{1,1} \preceq \begin{pmatrix} 12 & \perp \\ 11 & 0 \end{pmatrix} \quad \bar{\mathcal{A}}_{1,1}\bar{\mathcal{W}}_1 \preceq \begin{pmatrix} 13 & \perp \\ 12 & 0 \end{pmatrix}.$$

Remark Brander briefly described in his thesis [6] how to incorporate re-encoding into the Beelen–Brander interpolation algorithm by dividing out common powers of $L(X)$ in the first s columns of $\mathcal{A}_{s,\ell}$. Here we construct instead $\bar{\mathcal{A}}_{s,\ell}$ where powers of $L(X)$ are also multiplied on the latter $\ell - s$ columns, since we need the simple recursions of bases of $\varphi(M_{s,\ell})$ which enables the micro-steps.

However, before applying module minimisation, we could divide away this common factor from those columns and just adjust the weights accordingly (i.e., multiplying $X^{k(t-s)}$ on the tth element of $\bar{\mathcal{W}}_\ell$); this will further reduce the complexity of the minimisation step. The micro-steps would then need to be modified; the simplest way to repair this is to multiply back the powers of $L(X)$ before applying a micro-step, and then remove them again afterwards. With a bit more care one can easily do this cheaper, though the details become technical.

In asymptotic terms, the computational complexity of the iterative interpolation method stays exactly the same with re-encoding as without it, since $O(n - \deg L) = O(n - k) = O(n)$ under the usual assumption of n/k being constant. The same is true for the original re-encoding scheme of Kötter–Vardy [15,13]. However, most of the polynomials that are handled in the matrix minimisation will be of much lower degree than without re-encoding; for relatively high-rate codes this will definitely be noticeable in real computation time.

In [13, Thm. 10], it was shown that the root-finding procedure of Roth–Ruckenstein [21], or its divide-&-conquer variant by Alekhnovich [1] can be directly applied to an interpolation polynomial in $\varphi(M_{s,\ell})$, so we can avoid to construct and work on the larger polynomial in $M_{s,\ell}$. Instead of finding $f(X)$, one will find the power series expansion of $f(X)/L(X)$. The fraction in reduced form can be retrieved from the power series expansion using Padé approximation: e.g. by the Berlekamp–Massey algorithm or by module minimising a certain 2×2 matrix. See e.g. [19, Section 2.5] for a general description of the latter. From the reduced fraction, $f(X)$ can be obtained by re-extending the fraction.

Interestingly, one can easily calculate that the orthogonality defects stays the same, i.e., $\mathrm{D}(\mathcal{C}^{\mathrm{I}}_{s,\ell+1}\mathcal{W}_{\ell+1}) = \mathrm{D}(\bar{\mathcal{C}}^{\mathrm{I}}_{s,\ell+1}\bar{\mathcal{W}}_{\ell+1})$, where $\bar{\mathcal{C}}^{\mathrm{I}}_{s,\ell+1}$ is the matrix corresponding to a micro-step of type I in the re-encoded version. The analogue equality holds for type II. This means that, roughly, the number of row operations carried out by the module minimisation algorithm is unchanged.

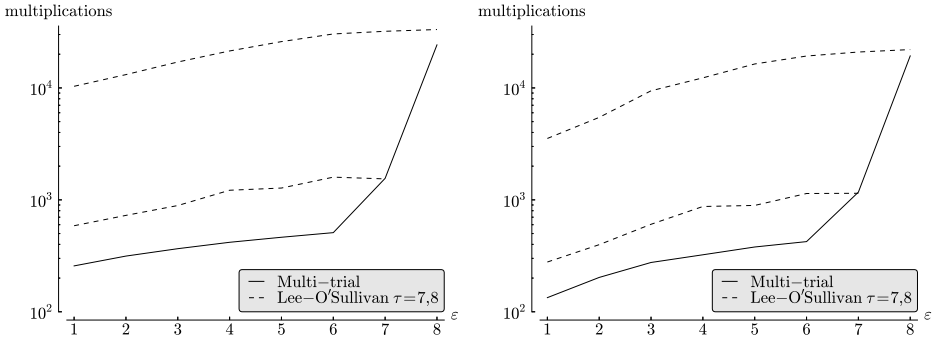
6 Simulation Results

The proposed algorithm has been implemented in Sage, Version 5.13 [22], using the Mulders–Storjohann algorithm for module minimisation, and the Roth–Ruckenstein root-finding procedure [21]. For comparison, we also implemented construction of $\mathcal{A}_{s,\ell}$, leading immediately to the Lee–O'Sullivan algorithm [16].

Figure 1 shows the total number of finite field multiplications performed for complete runs of the decoding algorithms, using the $\mathcal{GRS}(16, 4)$ code over $\mathbb{F}_{17}$ as considered in Example 3. For each algorithm, and for each number of errors $\varepsilon \leq \tau$, 1000 random codewords were generated and subjected to a random error pattern of weight precisely ε. The solid line gives the number of operations of the proposed multi-trial algorithm for any number of errors. For the Lee–O'Sullivan decoder, one chooses the maximal decoding radius initially, and the figure depicts choosing both $\tau = 7, 8$ as dashed lines. The minimum-distance choice of $\tau = 6$ coincides completely with the multi-trial algorithm since $\mathcal{A}_{s,\ell}$ for $\varepsilon \leq 6$ and so is not shown.

The figure demonstrate that the multi-trial algorithm provides a huge gain whenever there are fewer errors than Lee–O'Sullivan's target, while not having a disadvantage in the matching case. The right-hand graph shows the complexity when using the re-encoding transformation, and we can observe a speedup of between 30% and 50%. The number of operations spent in constructing the matrices and for the root-finding step are relatively small compared to the number of operations needed for the row-reduction (for the $\mathcal{GRS}(16, 4)$ code, less than 5%).

Unfortunately, the performance of our operation-counting implementation for the decoding algorithm does not allow to run simulations with much larger codes. Without counting, however, we can use optimised data structures in Sage [22] which run much faster. We have observed for the short $\mathcal{GRS}(16, 4)$ code that the system-clock time



(a) Without Re-Encoding (b) With Re-Encoding

Fig. 1 Comparison of the number of field multiplications for list decoding of an $\mathcal{GRS}(16, 4)$ code over $\mathbb{F}_{17}$. The operations of the Lee–O'Sullivan [16] algorithm with different aimed decoding radii and the proposed multi-trial algorithm are illustrated. Note the logarithmic y-axis. Subfigure (a) illustrates the number of operations without re-encoded points. The gain due to the re-encoding procedure is visible in Subfigure (b).

spent with these data structures correspond very well to the operations counted. Extrapolating, we can, albeit with larger uncertainty, make comparisons on larger codes using system-clock measurements. Doing so, we have observed behaviour resembling that of the $\mathcal{GRS}(16, 4)$ code. For example, decoding a $\mathcal{GRS}(64, 25)$ code up to 23 errors (requiring $(s, \ell) = (4, 6)$) showed the multi-trial being slightly faster than Lee–O'Sullivan in the worst case, while of course still giving a large improvement for fewer errors. When decoding a $\mathcal{GRS}(255, 120)$ code up to 74 errors (requiring $(s, \ell) = (4, 5)$), Lee–O'Sullivan [16] was slightly faster by about 15% in the worst case. It should also be noted that root-finding took up significantly more time for these larger codes, around 15% of the total time for Lee–O'Sullivan and 20% for the multi-trial.

As with any simulation, there are caveats to these results. In truth, only the wall-clock time spent by highly optimised implementations of various approaches can be fairly compared. Also, we did not test the asymptotically fast Alekhnovich's method for module minimisation. Investigations performed by Brander in Magma indicated that the gain in using this algorithm in place of Mulders–Storjohann might only be present once the code length exceeds about 4000 [6].

The implementation of the algorithm, including the simulation setup, is freely available via `http://jsrn.dk/code-for-articles`.

7 Conclusion

An iterative interpolation procedure for list decoding GRS codes based on Alekhnovich's module minimisation was proposed and shown to have the same worst-case complexity as Beelen and Brander's [3]. We showed how the target module used in Beelen–Brander can be minimised in a progressive manner, starting with a small module and systematically enlarging it, performing module minimisation in each step. The procedure takes advantage of a new, slightly more fine-grained complexity analysis of Alekhnovich's algorithm, which implies that each of the module refinement steps runs fast.

We furthermore incorporated the re-encoding transformation of Kötter and Vardy [15] into our method, which provides a noticeable, if not asymptotic, gain in computational complexity.

The main advantage of the algorithm is its granularity which makes it possible to perform fast multi-trial decoding: we attempt decoding for progressively larger decoding radii, and therefore find the list of codewords closest to the received. This is done without a penalty in the worst case but with an obvious benefit in the average case.

The Beelen–Brander approach for interpolation is not the asymptotically fastest: using the module minimisation algorithm by Giorgi *et al.* [10], one gains a factor ℓ. By a completely different approach, Chowdhury et al. [8] further beat this by a factor ℓ/s, achieving $O(\ell^2 s^2 n \log^{O(1)}(n))$. It is unclear for which sizes of the parameters these asymptotic improvements have concrete benefits, and whether a multi-trial approach can be developed for them.

Acknowledgement

The authors thank Daniel Augot for fruitful discussions. This work has been supported by German Research Council Deutsche Forschungsgemeinschaft (DFG) under grant Bo867/22-1 and Ze1016/1-1.

References

1. Alekhnovich, M.: Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed–Solomon Codes. IEEE Trans. Inform. Theory **51**(7) (2005)
2. Bassalygo, L.A.: New upper bounds for error-correcting codes. Probl. Inf. Transm. **1**(4), 41–44 (1965)
3. Beelen, P., Brander, K.: Key equations for list decoding of Reed–Solomon codes and how to solve them. J. Symbolic Comput. **45**(7), 773–786 (2010)
4. Beelen, P., Høholdt, T., Nielsen, J.S.R., Wu, Y.: On Rational Interpolation-Based List-Decoding and List-Decoding Binary Goppa Codes. IEEE Trans. Inform. Theory **59**(6), 3269–3281 (2013)
5. Bernstein, D.J.: List Decoding for Binary Goppa Codes. In: IWCC, pp. 62–80 (2011)
6. Brander, K.: Interpolation and List Decoding of Algebraic Codes. Ph.D. thesis, Technical University of Denmark (2010)
7. Cassuto, Y., Bruck, J., McEliece, R.: On the Average Complexity of Reed–Solomon List Decoders. IEEE Trans. Inform. Theory **59**(4), 2336–2351 (2013)
8. Chowdhury, M.F.I., Jeannerod, C.P., Neiger, V., Schost, E., Villard, G.: Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations. arXiv:1402.0643 (2014). URL `http://arxiv.org/abs/1402.0643`
9. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge Univ Press (2003)
10. Giorgi, P., Jeannerod, C., Villard, G.: On the Complexity of Polynomial Matrix Computations. In: Proceedings of International Symposium on Symbolic and Algebraic Computation, pp. 135–142. ACM (2003)
11. Guruswami, V., Sudan, M.: Improved Decoding of Reed–Solomon Codes and Algebraic Geometry Codes. IEEE Trans. Inform. Theory **45**(6), 1757–1767 (1999)
12. Johnson, S.M.: A New Upper Bound for Error-Correcting Codes. IEEE Trans. Inform. Theory **46**, 203–207 (1962)
13. Kötter, R., Ma, J., Vardy, A.: The Re-Encoding Transformation in Algebraic List-Decoding of Reed–Solomon Codes. IEEE Trans. Inform. Theory **57**(2), 633–647 (2011)
14. Kötter, R., Vardy, A.: A Complexity Reducing Transformation in Algebraic List Decoding of Reed–Solomon Codes. In: IEEE Information Theory Workshop (ITW), pp. 10–13. Paris, France (2003)
15. Kötter, R., Vardy, A.: Algebraic Soft-Decision Decoding of Reed-Solomon Codes. IEEE Trans. Inform. Theory **49**(11), 2809–2825 (2003)

16. Lee, K., O'Sullivan, M.E.: List Decoding of Reed–Solomon Codes from a Gröbner Basis Perspective. J. Symbolic Comput. **43**(9), 645–658 (2008)
17. Lenstra, A.: Factoring Multivariate Polynomials over Finite Fields. J. Comput. Syst. Sci. **30**(2), 235–248 (1985)
18. Mulders, T., Storjohann, A.: On Lattice Reduction for Polynomial Matrices. J. Symbolic Comput. **35**(4), 377–401 (2003)
19. Nielsen, J.S.R.: List decoding of algebraic codes. Ph.D. thesis, Technical University of Denmark (2013)
20. Nielsen, J.S.R., Zeh, A.: Multi-Trial Guruswami–Sudan Decoding for Generalised Reed–Solomon Codes. In: Workshop on Coding and Cryptography (2013)
21. Roth, R., Ruckenstein, G.: Efficient Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance. IEEE Trans. Inform. Theory **46**(1), 246 –257 (2000)
22. Stein, W., et al.: Sage Mathematics Software (Version 5.13, Release Date: 2013-12-15). The Sage Development Team (2013). `http://www.sagemath.org`
23. Sudan, M.: Decoding of Reed–Solomon Codes beyond the Error-Correction Bound. J. Complexity **13**(1), 180–193 (1997)
24. Tang, S., Chen, L., Ma, X.: Progressive List-Enlarged Algebraic Soft Decoding of Reed-Solomon Codes. IEEE Commun. Lett. **16**(6), 901–904 (2012)
25. Zeh, A., Gentner, C., Augot, D.: An Interpolation Procedure for List Decoding Reed–Solomon Codes Based on Generalized Key Equations. IEEE Trans. Inform. Theory **57**(9), 5946–5959 (2011)