Power Decoding Reed–Solomon Codes Up to the Johnson Radius

Nielsen, Johan Sebastian Rosenkilde

Published in:
I E E E Transactions on Information Theory

Publication date:
2015

Document Version
Preprint (usually an early version)

Link to publication

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Power Decoding Reed–Solomon Codes Up to the Johnson Radius

Johan S. R. Nielsen

Abstract

Power decoding, or “decoding using virtual interleaving” is a technique for decoding Reed–Solomon codes up to the Sudan radius. Since the method’s inception, it has been an open question if it possible to incorporate “multiplicities”, the parameter allowing the Guruswami–Sudan algorithm to decode up to the Johnson radius. In this paper we show how this can be done, and describe how to efficiently solve the resulting key equations. We investigate its failure behaviour theoretically as well as giving simulation results, and we show how the method can be made practically faster using the re-encoding technique or a syndrome formulation.

I. INTRODUCTION

Power decoding was originally proposed by Schmidt, Sidorenko and Bossert for low-rate Reed–Solomon codes (RS) [24]. Using shift-register synthesis techniques, the method allows to decode as many errors as the Sudan algorithm [29]. As opposed to Sudan’s list decoder, Power decoding returns at most one codeword but will in some cases simply fail. For random errors, this seem to occur with only very small probability, however.

The Sudan decoder generalises to the Guruswami–Sudan decoder [11] by introducing the multiplicity parameter, improving the decoding radius for all rates, allowing it to decode up to the Johnson bound. Since [24], it has been an open question whether it is likewise possible to introduce a “multiplicity parameter” into Power decoding and thereby increase the decoding radius up to the Johnson bound.

In this work we show how this can be done. The overall behaviour of the decoder is similar to Power decoding:

1) The equations are of a generalised shift-register type, and no root-finding as in Guruswami–Sudan is necessary.
2) The decoding radius becomes almost exactly that of the Guruswami–Sudan decoder (under the same choices of parameters).
3) There remains a low but non-zero probability of failing whenever one decodes beyond half the minimum distance.

Furthermore, we will show how to realise the decoder efficiently, yielding a complexity \( O^*(\ell^s n) \), where \( \omega \) is the exponent of matrix multiplication, and \( s, \ell \) are the multiplicity, respectively powering parameters of the decoder. This is very close to the best known complexities \( O^*(\ell^{s^{-1}} s^2 n) \) for the Guruswami–Sudan algorithm or the Wu list decoder [5].

In the next section we give an introduction to the previous key equation-based decoding algorithms: half-the-minimum distance and Power decoding. In Section III we then derive the new key equations. These are non-linear relations between polynomials which would allow us to decode, but it is non-trivial how to use this for efficient decoding. We describe this in Section IV using lattice basis reduction techniques and attaining the aforementioned complexity. The initial approach is refined to a slightly faster one in Section IV-A.

The improvements of Section IV-A also allows a simple way to derive a decoding radius bound which we do in Section V. This immediately gives a correspondence to the decoding radius of the Guruswami–Sudan algorithm. Power decoding will fail on certain received words even within this radius, however, and we investigate this in Section VI specifically, we show that the failure behaviour depends only on the error and not on the sent codeword. We then show that decoding always succeeds up to half the minimum distance, as well as bounding the failure probability for the case \( s = 2 \) and \( \ell = 3 \). Note that for the original Power decoding, analytic bounds on the failure probability have been obtained only when the powering degree is 2 or 3 [18], [25], [34]. In Section VII we give simulation results: these strongly back up the decoding capabilities of the algorithm.

In Section VIII we describe how the re-encoding technique of Kötter and Vardy [14] can be applied to the new Power decoding for reducing the practical complexity, if not the asymptotic one. In Section IX we similarly show how our Gao-type key equations can be rewritten into syndrome-type ones, resulting in a similar complexity reduction.

The decoding method has been implemented in Sage v. 6.6 [28] and can be downloaded from http://jsrn.dk/code-for-articles, together with the code for running the simulation.

II. PRELIMINARIES AND EXISTING KEY EQUATIONS

A. GRS codes

Consider some finite field \( \mathbb{F} \). Choose \( n \leq |\mathbb{F}| \) as well as distinct \( \alpha_1, \ldots, \alpha_n \in \mathbb{F} \) as well as non-zero (not necessarily distinct) \( \beta_1, \ldots, \beta_n \in \mathbb{F} \). For any \( f \in \mathbb{F}[x] \) we write

\[
ev(f) = (\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)).
\]
The \([n,k,d]\) Generalised Reed-Solomon (GRS) code is the set
\[ C = \{ \text{ev}(f) \mid f \in \mathbb{F}[x], \ deg f < k \}. \]

The \(\alpha_i\) are called evaluation points and the \(\beta_i\) column multipliers. \(C\) has minimum distance \(d = n - k + 1\) and the code is therefore MDS.

Consider now that some \(e = (e_1, \ldots, e_n)\) was sent with \(e = \text{ev}(f)\) for some \(f \in \mathbb{F}[x]\), and that \(r = (r_1, \ldots, r_n) = e + e\) was the received word with error \(e\).

Note that column multipliers can be ignored in decoding: we simply compute \(r' = (r_1/\beta_1, \ldots, r_n/\beta_n) = e' + e'\), where \(e'\) is in the code \(C'\) which has the same evaluation points \(\alpha\) but where all \(\beta_i = 1\). \(e'\) is an error vector with the same number of errors as \(e\). In the remainder of the article, we therefore assume \(\beta_i = 1\).

Introduce two essential polynomials, immediately computable by the receiver:
\[ G = \prod_{i=1}^n(x - \alpha_i) \quad R : \deg R < n, \ R(\alpha_i) = r_i, \ i = 1, \ldots, n \]

\(G\) can be pre-computed, while \(R\) is computed upon receiving \(r\) using Lagrange interpolation.

As usual for key equation decoders, the algorithm will revolve around the notion of error locator \(\Lambda\) and error evaluator \(\Omega\):
\[ \Lambda = \prod_{i \in \mathcal{E}}(x - \alpha_i) \quad \Omega = -\sum_{i \in \mathcal{E}} e_i \zeta_i \prod_{j \notin \mathcal{E}}(x - \alpha_j) \]

where \(\zeta_i = \prod_{j \neq i} (\alpha_i - \alpha_j)^{-1}\). Note that \(\epsilon = \deg \Lambda > \deg \Omega\).

These four polynomials are related by the following relation, which will be at the centre of our investigations:

**Lemma 1:** \(\Lambda(f - R) = \Omega G\)

**Proof:** The closed formula for Lagrange interpolation implies that \(f - R = \sum_{i=1}^n -e_i \zeta_i \prod_{j \neq i}(x - \alpha_j)\). This directly means
\[ \Lambda(f - R) = \Lambda \sum_{i \in \mathcal{E}} -e_i \zeta_i \prod_{j \neq i}(x - \alpha_j) = \sum_{i \in \mathcal{E}} -e_i \zeta_i \left( \frac{\Lambda}{x - \alpha_i} \right) G = \Omega G \]

The objects \(c, r, e, \Lambda, \Omega\), etc. introduced here will be used in the remainder of the article.

In complexity discussions, we count arithmetic operations in the field \(\mathbb{F}\). We will use \(\omega\) as the exponent for matrix multiplication, i.e. \(2 \leq \omega \leq 3\). We use \(O(\cdot)\) as big-\(O\) but ignoring log-factors. In a few places we also use \(M(n)\) to denote the complexity of multiplying together two polynomials; we can trivially use \(M(n) \in O(n^2)\) or we can have \(M(n) \in O^*(n)\), see e.g. [31].

### B. Classical Key Equations

Let us revisit the key equation implicit in Gao’s decoder [8], which follows directly from Lemma [1]

\[ \lambda R \equiv \Lambda f \mod G \quad (1) \]

This is a non-linear equation in the unknowns \(\Lambda\) and \(f\), and it is not immediately obvious how to build an efficient decoder around it. The good idea is to ignore the non-linear relation: we replace the sought quantities \(\Lambda\) and \(\Lambda f\) with unknowns \(\lambda\) and \(\psi\), both in \(\mathbb{F}[x]\), and such that
\[ \lambda R \equiv \psi \mod G . \]

This is now a linear relation, but unfortunately with infinitely many solutions. We further restrict the solutions by requiring
\[ \deg \lambda + k - 1 \geq \deg \psi \].

Note that this is satisfied if \(\lambda\) is replaced by \(\Lambda\) and \(\psi\) by \(\Lambda f\). Finally, we seek such \(\lambda, \psi\) where \(\lambda\) is monic and has minimal degree. The hope is now that \(\lambda = \Lambda\) even though we solved for a much weaker relation than (1); effectively, it is therefore the low degree of \((\lambda R \mod G)\) which is used to solve for \(\Lambda\). Solving such requirements for \(\lambda\) and \(\psi\) is sometimes known as rational function reconstruction [31]. They are easy to solve for in complexity \(O(n^2)\) or \(O^*(n)\), using e.g. the extended Euclidean algorithm [7, 8, 30].

It can be shown that whenever \(\epsilon < d/2\) we get \(\lambda = \Lambda\) and \(\psi = \Lambda f\), see e.g. [8]. Then \(f = \psi/\lambda\) and decoding is finished. However, whenever \(\epsilon \geq d/2\), the approach will essentially never work.

Whenever 0 is not an evaluation point, i.e. \(\alpha_i \neq 0\) for all \(i\), then the equation can be rewritten to the more classical syndrome key equation [4]. First some notation: for \(p \in \mathbb{F}[x]\), let \([d]p\) denote the reversal of the coefficients of \(p\) at degree \(d\), i.e. \([d]p = x^dp(x^{-1})\) for some integer \(d \geq \deg p\). To lighten the notation, we will often omit the \([d]\) when there is an implied upper bound on the degree of the polynomial being reversed; to be precise, note that we reverse on the upper bound on the degree, and not on the actual degree which might happen to be lower.
Introduce $S(x)$ as the power series expansion of $\frac{R}{G}$ truncated at $x^{n-k}$. Then by reversing Lemma 1 at degree $n-1+\epsilon$ we get:

$$\Lambda R = \Lambda f - \Omega G$$

$$\frac{[\epsilon+n-1]}{[\Lambda R]} = \frac{[\epsilon+k-1]}{(\Lambda f)x^{n-k}} - \frac{[n+\epsilon-1]}{(\Omega G)}$$

Since $x \nmid G$ this implies the well-known formula:

$$\Lambda S \equiv \Omega \mod x^{n-k}$$

A (now less obvious) algebraic relation exist between $\Lambda$ and $\Omega$. To allow for efficient solving, we forget this relation, and replace $\Lambda$ and $\Omega$ by unknowns $\bar{\lambda}$ and $\bar{\omega}$, and solve for the minimal degree $\bar{\lambda}$ satisfying

$$\bar{\lambda} S \equiv \bar{\omega} \mod x^{n-k}$$

This time the modulus is a power of $x$; solving such an equation for $\bar{\lambda}$ and $\bar{\omega}$ is known as Padé approximations [2] or a linear feedback shift-register [22, Section 6.7]. It can be solved in complexity $O(n^2)$ or $O(n)$ using either the extended Euclidean algorithm or the Berlekamp–Massey algorithm.

One can again show that this approach will succeed, i.e. in the end $\bar{\lambda} = \bar{\lambda}$, whenever $\epsilon \leq \lfloor (d-1)/2 \rfloor$ [4]. Slightly stronger, one can show that the approach will succeed if and only if the Gao key equation approach succeeds [18].

C. Simply Powered Key Equations

Power decoding, or decoding by virtual interleaving [25], is a generalisation of (1) where not one but multiple non-linear relations between $\Lambda$ and $f$ are identified. The original formulation of [25] is based on the classical syndrome key equation, while powering the Gao key equation was described in [18]. We will begin with the latter. Using Lemma 1 one can easily prove that:

$$\Lambda R^t \equiv f^t \mod G, \quad t = 1, 2, \ldots .$$

We will give a more general statement in the next statement so omit the proof here.

Again this gives non-linear relations between $\Lambda$ and $f$. To solve them efficiently, we use only the first $\ell$ of the equations, for some chosen $\ell$, and introduce unknowns $\lambda, \psi_1, \ldots, \psi_\ell \in \mathbb{F}[x]$. We then solve for $\lambda, \psi_i$ such that $\lambda$ is monic and of minimal degree such that

$$\lambda R^t \equiv \psi_t \mod G, \quad t = 1, \ldots, \ell$$

Finally, we hope that the found $\lambda = \Lambda$. In this case $f = \psi_1/\lambda$.

Notice that this linearisation process immediately renders key equations for large enough $t$ useless: when $\deg \lambda + \ell(k-1) \geq n$ then any choice of $\lambda$ will satisfy the $t$'th key equation simply setting $\psi_t = (\lambda R \mod G)$. That gives the rough bound $\ell < n/(k-1)$.

By regarding the linearised problem as a linear system of equations, and counting available coefficients versus constraints, one arrives at an expression for the greatest number of errors we should expect to be decodable:

$$\epsilon \leq \ell \cdot \frac{n-1}{\ell+1} - \frac{\ell}{\ell+1}$$

This argument does not imply that we will necessarily succeed when the bound is satisfied: decoding success follows if the constructed linear system has full rank, but this is not always the case. That means that for rare cases, decoding might fail for fewer errors than $\epsilon$. Bounding the probability that this occurs has proven difficult. We now know upper bounds when $\ell = 2, 3$ [18], [25], and Schmidt, Sidorenko, and Bossert posed a conjecture, backed by simulation, on the probability in general [25].

Equation (3) is concave in $\ell$, and its integral maxima suggests the value of $\ell$ one should choose to maximise the decoding radius. Analysis reveals that whenever $k/n > 1/3$, one should simply choose $\ell = 1$, i.e. classical key equation decoding. Thus Power decoding is only useful for low-rate codes. Note that (3) is almost the same bound as the Sudan decoding algorithm [29], which is the Guruswami–Sudan with multiplicity 1.

The syndrome variant of (3) was again historically the first [24]: define $S^{(t)}$ as the power series expansion of $\frac{R^{(t)}}{G}$ truncated at $x^{n-(t(k-1)-1)}$, where $R^{(t)}$ is the unique polynomial of degree less than $n$ such that $R^{(t)} \equiv R^t \mod G$. Then one can easily show [18], using the same rewriting as in Section II-B

$$\Lambda S^{(t)} \equiv \Omega \mod x^{n-(t(k-1)-1)},$$

1By inserting the explicit Lagrange interpolation formula for $R$, it is easy to see that this definition of the syndrome polynomial corresponds to the classical one, in e.g. [21, Section 6.2].
where $\Omega_t$ are certain polynomials of degree at most $\epsilon - 1$ that we omit defining explicitly. It can be shown using the same rewriting that Power syndrome decoding fails if and only if Power Gao decoding fails [18].

For the Gao formulation, the linearised problem to solve is sometimes known as vector rational function reconstruction [20], and for the syndrome formulation as simultaneous Padé approximation [2] or multi-sequence shift-register [25]. In the latter case, see [25, 27] for an $O(n^2)$ algorithm, and [26] for an $O((\ell^\omega)n)$ algorithm. For the Gao formulation, we need the more general case considered in [16], which gives multiple algorithms, with complexities $O(\ell^2 n^2)$, $O((\ell^\omega)n)$ or $O((\ell nT)$, where $T$ depends on the sparseness of $G$ and is between 1 and $O(\log n)$. In particular, if the GRS code evaluates at all elements of the field then $G = x^n - x$ and $T = 2$. The approach in [16] is based on computing reduced bases of carefully selected $\mathbb{F}_q[x]$ modules. What we describe in Section IV to solve the new Powered key equations is a generalisation of this approach.

### III. New Key Equations

In this section we describe the main result of the paper, namely a new generalisation of Power decoding where we introduce a second parameter, the multiplicity. The resulting relations will again be non-linear in $\Lambda$ and $f$, and we will employ a similar linearisation strategy. We will see in Section IV how the linear problem can be solved efficiently using a lattice basis reduction approach.

The generalised key equations are described in the following theorem:

**Theorem 2:** For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then

\[
\Lambda^s f^t \equiv \sum_{i=0}^{s-1} (\Lambda^{s-i} \Omega^i) \left( \binom{t}{i} R^{t-i} G^i \right) \quad \mod G^s
\]

for $t = s, \ldots, \ell$

**Proof:** We simply rewrite

\[
\Lambda^s f^t = \Lambda^s (R + (f - R))^t = \sum_{i=0}^{t} \binom{t}{i} \Lambda^s (f - R)^i R^{t-i}
\]

If $t < s$ then $\Lambda^s (f - R)^i = \Lambda^{s-i} \Omega^i G^i$ for each $i$ by Lemma [1]. This finishes the first part of the theorem.

If $t \geq s$ then for $i = s, \ldots, \ell$, the summand equals $\binom{t}{i} \Lambda^{s-i} \Omega^i G^i R^{t-i}$ due to Lemma [1] which is 0 modulo $G^s$. Replacing $\Lambda^s (f - R)^i$ by $\Lambda^{s-i} \Omega^i G^i$ for $i < s$ as before gives the sought.

The above theorem describes $\ell$ equations in the unknowns $\Lambda^s, \Lambda^{s-1} \Omega, \ldots, \Lambda \Omega^{s-1}$ as well as $\Lambda^s f, \ldots, \Lambda^s f^\ell$. These are “key equations” in the following sense: the inner product of the first set of unknowns with a vector of known polynomials (the $\binom{t}{i} R^{t-i} G^i$) have surprisingly low degree – either immediately or reduced modulo $G^s$ – since it is the degree of $\Lambda^s f^t$.

As with the previous key equation decoding algorithms described in Section II, we perform the following linearisation to make the problem of finding $\Lambda$ and $f$ tractable:

**Problem 3:** Find a vector $(\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell) \in \mathbb{F}[x]^{s+\ell}$ with $\lambda_0$ monic and such that the following requirements are satisfied:

1a) $\psi_t = \sum_{i=0}^{t} \lambda_i \left( \binom{t}{i} R^{t-i} G^i \right)$, for $t = 1, \ldots, s - 1$

1b) $\psi_t \equiv \sum_{i=0}^{s-1} \lambda_i \left( \binom{t}{i} R^{t-i} G^i \right) \mod G^s$, for $t = s, \ldots, \ell$

2) $\deg \lambda_0 \geq \deg \lambda_i + i$, for $i = 1, \ldots, s - 1$

3) $\deg \lambda_0 \geq \deg \psi_t - t(k - 1)$, for $t = 1, \ldots, \ell$

Clearly $\Lambda = (\Lambda^s, \Lambda^{s-1} \Omega, \ldots, \Lambda \Omega^{s-1}, \Lambda^s f, \ldots, \Lambda^s f^\ell)$ satisfies these requirements, but there are unfortunately infinitely many other vectors satisfying them. We will therefore seek the one of least degree, i.e., where $\deg \lambda_0$ is minimal; the hope is then that this vector is $\Lambda$. In that case, decoding will be completed simply by computing $f = \psi_1/\lambda_0$.

Note that as in the simpler Power decoding of Section II-C the above linearisation implies a rough bound for the choices of $\ell$, namely $\ell < sn/(k - 1)$. For $t \geq sn/(k - 1)$, whatever the values of $\lambda_i$, we can choose $\psi_t$ to satisfy item 1b of Problem 3 and have degree less than $sn$, and item 3 is then trivially satisfied.
IV. SOLVING THE KEY EQUATIONS

We will now show how one can use \( \mathbb{F}[x] \)-lattice basis reduction to find a minimal solution to Problem 3. This approach is very closely related to that of \cite{16} for solving the powered key equations of Section II-C. This, in turn, lends much from the Gröbner basis description for classical key equation solving by Fitzpatrick \cite{7}.

To solve Problem 3, consider first \( \mathcal{M} \) as the space of vectors \((\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_{\ell}) \in \mathbb{F}[x]^{s+\ell}\) just satisfying requirements 1a and 1b. Clearly \( \Lambda \in \mathcal{M} \). It turns out that \( \mathcal{M} \) is a free \( \mathbb{F}[x] \) module and in fact we know a basis for it:

**Proposition 4:** \( \mathcal{M} = \text{Row}_{\mathbb{F}[x]}(M) \), the \( \mathbb{F}[x] \)-row space of \( M \in \mathbb{F}[x]^{(\ell+1)\times(s+\ell)} \), where

\[
M = \begin{bmatrix}
I_{s\times s} & N \\
0_{(\ell-s+1)\times s} & 0_{(\ell-s+1)\times (s-1)} & G^sI_{(\ell-s+1)\times(\ell-s+1)}
\end{bmatrix},
\]

where \( N \in \mathbb{F}[x]^{s\times \ell} \) is the matrix whose \((i, t)\)th entry is

\[
N[i, t] = \left( \ell \atop t \right) R^{t-i}G^i \mod G^s, \quad i = 0, \ldots, s-1 \text{ and } t = 1, \ldots, \ell,
\]

that is,

\[
N = \begin{bmatrix}
R & R^2 & \ldots & R^{s-1} & \ldots & R^\ell \\
0 & 2RG & \ldots & (s-1)R^{s-1}G & \ldots & (\ell)R^{s-1}G \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \ldots & G^{s-1} & \ldots & (s-1)RG^{s-1}
\end{bmatrix} \mod G.
\]

**Proof:** Let \( m_j \) denote the rows of \( M \) for \( j = 0, \ldots, \ell - 1 \). To show \( \mathcal{M} \supset \text{Row}_{\mathbb{F}[x]}(M) \), simply note that each \( m_j \) is in \( \mathcal{M} \); for \( j < s \) then \( m_j \) corresponds in the equations of 1a and 1b to setting \( \lambda_i = 0 \) for all \( i \neq j \), \( \lambda_j = 1 \), as well as \( \psi_i = (\ell)R^{t-i}G^i \) for \( t = 0, \ldots, \ell \). This clearly leaves them satisfied. For \( j \geq s \), \( m_j \) corresponds to setting \( \lambda_i = \psi_i = 0 \) for all \( i \) and and \( \lambda_j = G^s \equiv 0 \mod G^s \).

Now for the other inclusion, leading to equality. For any \( v = (\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_{\ell}) \in \mathcal{M} \) then the vector \( v' = \sum_{i=0}^{s-1} \lambda_i m_i \) agrees with \( v \) in the first \( 2s - 1 \) positions. For the remaining positions \( j = s, \ldots, \ell \) of \( v' \), it is congruent to \( \psi_j \mod G^s \). Therefore there exists \( q_0, \ldots, q_{\ell} \in \mathbb{F}[x] \) such that \( v = v' + \sum_{i=s}^{\ell} q_i m_i \), and thus \( \mathcal{M} \subset \text{Row}_{\mathbb{F}[x]}(M) \).

To find a minimal solution to Problem 3 we should therefore seek a vector \( v = (\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_{\ell}) \in \text{Row}_{\mathbb{F}[x]}(M) \) such that:

i) \( \deg \lambda_0 \geq \deg \lambda_i + i \)

ii) \( \deg \lambda_0 + t(k-1) \geq \deg \psi_i \)

iii) \( \deg \lambda_0 \) is minimal under these constraints and \( \lambda_0 \) is monic.

These goals turn out to be achievable by finding another matrix whose rows span \( \mathcal{M} \) but which is in weak Popov form. This form was introduced by Mulders and Storjohann in \cite{15} as a slightly stronger form than row reduced \cite{12} p. 380, which also exactly allows to argue about restrictions such as the degree inequalities above. The rows of a matrix in weak Popov is also a Gröbner basis for the module \( \mathcal{M} \) for the term-over-position ordering; however we will stay with the matrix language in this exposition. Our strategy is very similar to finding short vectors in modules by computing a row reduced basis, see e.g. \cite{31} Problem 16.12. In this settings, shifts as we will use have also been considered, e.g. \cite{35}.

**Definition 5:** The leading position of a non-zero vector \( v \in \mathbb{F}[x]^m \), written \( \text{LP}(v) \), is the right-most entry in \( v \) with maximal degree among the entries of \( v \). A matrix \( V \in \mathbb{F}[x]^{m_1 \times m_2} \) is in weak Popov form if the leading positions of the non-zero rows are all different.

**Proposition 6:** Let \( V \in \mathbb{F}[x]^{m_1 \times m_2} \) be a basis in weak Popov form of a module \( \mathcal{V} \). Any non-zero \( b \in \mathcal{V} \) satisfies \( \deg v \leq \deg b \) where \( v \) is the row of \( V \) with \( \text{LP}(v) = \text{LP}(b) \). If a leading position is not represented by a row in \( V \), then no vector in \( \mathcal{V} \) has that leading position.

**Proof:** Let \( u \in \mathcal{V} \) be non-zero, and so there exists \( a_0, \ldots, a_{\ell} \in \mathbb{F}[x] \) not all zero such that \( u = \sum_{i=0}^{\ell} a_i v_i \) where the \( v_i \) are the rows of \( V \). The \( v_i \) all have different leading position, so the \( a_i v_i \) also have different leading position among those \( i \) where \( a_i \neq 0 \). Note that for any two \( u_1, u_2 \) with \( \text{LP}(u_1) \neq \text{LP}(u_2) \), then \( u_1 + u_2 \) has the same degree and leading position of either \( u_1 \) or \( u_2 \). Applied inductively, that implies that there is an \( i \) such that \( \text{LP}(u) = \text{LP}(a_i v_i) \) and \( \deg u = \deg(a_i v_i) \geq \deg v_i \).

The above proposition means that if \( V \) is a basis in weak Popov form of some module \( \mathcal{V} \), then the rows of \( V \) have minimal degree for each possible leading position. So we can use the weak Popov form to find small-degree vectors which has the greatest degree polynomial on a specific index. Our degree restrictions single out \( \lambda_0 \) as somehow “leading”, but under integral shifts, e.g. of the form \( \deg (\lambda_0 + t(k-1)) \geq \deg \psi_i \). We will handle these shifts by incorporating them directly into the module.
First, we introduce non-negative variables $\mu_0, \ldots, \mu_{s-1}, \eta_1, \ldots, \eta_t \in \mathbb{N}_0$ as
\[
\mu_0 = 1 + \ell(k-1) \quad \mu_i = i + \ell(k-1), \; i > 0 \quad \eta_t = (\ell-t)(k-1), \; \forall t.
\] (5)

Our degree restrictions now read
\[
\deg \lambda_0 + \mu_0 > \deg \lambda_i + \mu_i, \quad i = 1, \ldots, s - 1
\]
\[
\deg \lambda_0 + \mu_0 > \deg \psi_i + \eta_t, \quad t = 1, \ldots, \ell
\] (6) (7)

Notice that a vector $v = (\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell)$ satisfies these degree restrictions if and only if $LP(vD) = 1$, where
\[
D = \text{diag}(x^{\mu_0}, \ldots, x^{\mu_{s-1}}, x^{\eta_1}, \ldots, x^{\eta_t}).
\] (8)

Consider therefore the module $\hat{M} = \{ vD \mid v \in M \}$, spanned by the rows of $M = MD$. We arrive at:

**Corollary 7:** Let $\hat{B} = BD$ be a basis of $\hat{M}$ and in weak Popov form, and let $\hat{v}$ be the row of $\hat{B}$ with leading position 1. Then $(\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell) = \gamma \hat{v} D^{-1}$ constitutes a solution to Problem 3 such that $\deg \lambda_0$ is minimal, where $\gamma \in \mathbb{F}^*$ is chosen such that $\lambda_0$ is monic.

**Proof:** By Corollary 7, $\hat{v}$ must have minimal degree among vectors in $\hat{M}$ with leading position 1. The above discussion then implies that $(\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell)$ satisfies the degree constraints of Problem 3 and that $\lambda_0$ has minimal degree among the first term of vectors satisfying these constraints. This vector also satisfies the congruence constraint of Problem 3 since it is in $M$.

Note that any basis of $\hat{M}$ seen as a matrix must be $\mathbb{F}[x]$-divisible on the right by $D$. So to find a $\hat{B}$ satisfying Corollary 7 we need only compute $\hat{B} \sim M$ such that $\hat{B}$ is in weak Popov form. By $\sim$ we mean unimodular equivalence, i.e. $A \sim B$ for two $A, B \in \mathbb{F}[x]^{m_1 \times m_2}$ if there exists an invertible matrix $U \in \mathbb{F}[x]^{m_1 \times m_2}$ such that $A = UB$. The $\mathbb{F}[x]$ row spaces of matrices $A$ and $B$ are the same if and only if $A \sim B$.

The complete decoding algorithm, with the weak Popov form computation as a black box, is given as Algorithm 1.

**Proposition 8:** Algorithm 1 is correct.

**Proof:** For any codeword $\hat{c} \in C$, there is an associated error $\hat{e} = r - \hat{c}$ and thus error locator $\hat{\Lambda}$ and error evaluator $\hat{\Omega}$. These satisfy Theorem 2 and therefore induce a solution to Problem 3. The first component of this solution is $\hat{\Lambda}^s$. By Corollary 7 the $(\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell)$ computed in Line 4 is a solution to Problem 3 where the first component $\lambda_0$ has minimal degree. No codeword can therefore have distance less than $\deg \lambda_0/s$ from $r$, since it would induce a solution to Problem 3 with smaller degree than $\deg \lambda_0$ on the first component.

If fail is not returned in Line 5 then the computed $f$ satisfies $\deg f < k$, since $\deg \psi_1 \leq \deg \lambda_0 + (k-1)$. Thus $ev(f) \in C$. Since $ev(f)$ is only returned if its distance to $r$ is exactly $\deg \lambda_0/s$, this must be a codeword of minimal distance to $r$.

Algorithm 1 leaves unspecified how to compute $\hat{B}$, i.e. how to compute a basis of $\hat{M}$ in weak Popov form. Since we are initially given a different basis of $\hat{M}$, namely $M$, the problem is that of finding a matrix which is unimodular equivalent to $M$ but in weak Popov form. This problem is well-studied in computer algebra, and several algorithms exist which solve this problem directly [1], [15] or through a related form [9], [10], [23], [35]. In particular, we have:

**Proposition 9:** Given a matrix $A \in \mathbb{F}[x]^{m_1 \times m_2}$ there exists an algorithm to compute a matrix $B \in \mathbb{F}[x]^{m_1 \times m_2}$ in weak Popov form and $B \sim A$ in complexity $O^*(m^2 M(\deg A))$ [10], as well as one for computing it in complexity $O(m^3 \deg A^2)$ [15], where $m = \max(m_1, m_2)$.

**Corollary 10:** Algorithm 1 can be implemented with asymptotic complexity either $O(\ell^2 sn)$ or $O(\ell^3 s^2 n^2)$.

**Proof:** $G^t$ can be precomputed for $i = 1, \ldots, s$. Computing $R$ can be done in $O(M(n))$ using Lagrange interpolation, see e.g. [31] p. 297. We compute all $R^t \mod G^t$ for $i = 1, \ldots, \ell$ in time $O(\ell s M(n))$. Constructing $\hat{M}$ then requires $O(\ell^2)$ elements that are scalings of the product of two known polynomials of degree at most $sn$, requiring $O(\ell^2 M(sn))$. Clearly, the remaining lines apart from Line 5 are cheaper. We have $\deg \hat{M} \in O(sn)$ since $\ell(k-1) < sn$. Thus the complexities for executing line Line 5 given by Proposition 9 dominates, whether or not a fast polynomial multiplication algorithm is used (i.e. whether setting $M(N) = O^*(N)$ or $M(N) = N^2$).

**Remark 11:** A realisation of the interpolation step of the Guruswami–Sudan having complexity $O^*(\ell^2 sn)$ was first proposed by Cohn and Heninger [6], and a similar approach is possible for obtaining this complexity with the Wu list decoder [3]. Chowdhury et al. [3] recently presented a slightly faster realisation for both list decoders, having complexity $O^*(\ell^2 s^2 n^2)$. Without fast arithmetic, the fastest interpolation in Guruswami–Sudan is by Gentner, Augot and Zeh [33] having complexity $O(\ell^4 s n^2)$. 
Algorithm 1 Efficient Power Decoding with Multiplicities

Input: \( \mathbf{r} \in \mathbb{F}_q^n, s, \ell \in \mathbb{Z}_+ \).

Output: \( \hat{c} \in \mathcal{C} \) such that \( \text{dist}(\hat{c}, \mathbf{r}) \) is minimal among codewords in \( \mathcal{C} \), or fail

1. Compute \( \mathbf{R} \) such that \( \mathbf{R}(r_i) = \mathbf{r}_i \).
2. Construct \( \mathbf{M} \) as in Proposition 6 and \( \hat{\mathbf{M}} \equiv \mathbf{MD} \), where \( \mathbf{D} \) is as in (8).
3. Compute \( \hat{\mathbf{B}} \) in weak Popov form such that \( \hat{\mathbf{B}} \sim \mathbf{M} \).
4. Let \( \hat{v} \) be the row of \( \hat{\mathbf{B}} \) with \( \text{lp}(\hat{v}) = 1 \). Let \( (A_0, \ldots, A_{s-1}, \psi_1, \ldots, \psi_\ell) \equiv \gamma \hat{v} \mathbf{D}^{-1}, \) with \( \gamma \in \mathbb{F}_q^* \) such that \( \lambda_0 \) is monic.
5. If \( \lambda_0 \) divides \( \psi_1 \), let \( f \equiv \psi_1/\lambda_0 \). Otherwise fail.
6. If \( \text{dist}(\mathbf{r}, \text{ev}(f)) = \deg \lambda_0/s \) then return \( \text{ev}(f) \). Otherwise fail.

### A. A Punctured Module by Ignoring Error-Evaluators

It is possible to obtain a smaller matrix than \( \mathbf{M} \) which will provide us with equally good decoding performance. This results in a faster decoding algorithm, though the asymptotic complexity remains unchanged. A second benefit of the optimisation is that it makes it easier for us to reason on the decoding radius of the algorithm in Section V.

The optimisation is based on two observations: Firstly, for decoding we do not need to know \( \lambda_1, \ldots, \lambda_s \) since the later steps of the algorithm only uses \( \lambda_0 \) and \( \psi_1 \). Secondly, in the failure probability bound that we derive in Section VI (for \( s = 2 \) and \( \ell = 3 \)) the degree restrictions \( \deg \lambda_0 > \deg \lambda_i \) are not used, and thus perhaps they have little influence on the failure probability in general.

In the lattice view, this means that we can simply delete columns 2 to \( s \) in both \( \mathbf{M} \) and \( \mathbf{D} \). Bringing this smaller matrix \( \hat{\mathbf{M}} \hat{\mathbf{D}} \in \mathbb{F}_q[x]^{(\ell+1) \times (\ell+1)} \) to weak Popov form will still solve for the remaining conditions. As a result, we obtain a vector \( (\lambda_0, \psi_1, \ldots, \psi_\ell) \) hopefully equal to \( (\lambda^0, \Lambda^s f, \ldots, \Lambda^s f^\ell) \).

Note that the punctured matrix \( \hat{\mathbf{M}} \) takes the form:

\[
\hat{\mathbf{M}} = \begin{bmatrix}
1 & R & R^2 & \cdots & R^{s-1} & R^s \\
0 & G & 2RG & \cdots & (s-1)R^{s-2}G & sR^{s-1}G \\
0 & 0 & G^2 & \cdots & (s-1)G^{s-3}G^2 & (s^2)R^{s-2}G^2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & G^{s-1} & sRG^{s-1} \\
& & & & & \\
0_{(\ell-s+1) \times s} & & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
\end{bmatrix}
\]

That is, \( \hat{\mathbf{M}} \) is a square, upper-diagonal matrix of full rank.

It is not theoretically clear whether working with \( \hat{\mathbf{M}} \) instead of \( \mathbf{M} \) could result in an increased failure probability. However simulations indicate this is not the case. Also, as already mentioned, the failure probability bound derived in Section VI applies for when working with \( \hat{\mathbf{M}} \).

### V. Decoding Radius

We are now in a position to discuss how many errors the method will usually cope with. When calling this a “decoding radius” we need to be wary; indeed, the method will fail for certain received words whenever the number of errors is at least \( d/2 \), and this is unavoidable since it is a unique decoding algorithm. Therefore, “decoding radius” really involves two parts: 1) how many errors should we at most expect to be able to correct; and 2) what is the probability that we will fail within this number of errors.

Perhaps not surprisingly, the latter question is much more difficult than the former. In this section we will derive an upper bound on error correction. In the next section we discuss the failure behaviour when fewer errors than this has occurred.

The decoding radius upper bound that we will derive is based on bringing \( \hat{\mathbf{M}} \hat{\mathbf{D}} \) from Section IV.A to weak Popov form. Recall that what we are essentially doing is finding the lowest degree vector in \( \text{Row}_{\mathbb{F}_q[x]}(\hat{\mathbf{M}}\hat{\mathbf{D}}) \) which has first position as leading. The outcome of imposing the leading position requirement is sensitively dependent on the module \( \mathcal{M} \), but bounding the size of the lowest degree vector overall is easy:

**Proposition 12:** A vector \( \mathbf{v} = (\lambda_0, \psi_1, \ldots, \psi_\ell) \) such that \( \mathbf{v} \hat{\mathbf{D}} \) is a minimal degree vector in \( \text{Row}_{\mathbb{F}_q[x]}(\hat{\mathbf{M}}\hat{\mathbf{D}}) \) satisfies

\[
\frac{\deg \lambda_0}{s} \leq \tau_{\text{row}}(s, \ell) \equiv \frac{2(\ell-s+1) + 1}{2(\ell+1)} \frac{\ell}{s(\ell+1)} - \frac{\ell}{2s} (k-1) - \frac{\ell}{s(\ell+1)}.
\]

If \( \epsilon > \tau_{\text{row}}(s, \ell) \) then \( \deg \Lambda > \deg \lambda_0/s \).

**Proof:** Let \( \hat{\mathbf{B}} \) be a matrix such that \( \hat{\mathbf{B}} \hat{\mathbf{D}} \) is in weak Popov form and unimodular equivalent to \( \hat{\mathbf{M}} \hat{\mathbf{D}} \). Recall that \( \hat{\mathbf{D}} = \text{diag}(x^{\mu_0}, x^{\mu_1}, \ldots, x^{\mu_\ell}) \).
We know from Proposition 6 that since $\hat{M}$ is square then for any leading position, $\hat{B}\hat{D}$ contains a vector of minimal degree in $\text{Row}_{\mathbb{F}[x]}(\hat{M}\hat{D})$ for that leading position, and therefore it also contains a row with minimal degree overall, i.e. with the same degree as $\nu\hat{D}$. Now, it is easy to see that since $\hat{B}\hat{D}$ is in weak Popov form, then $\deg \det(\hat{B}\hat{D}) = \sum_{i=1}^{\ell+1} \deg \hat{b}_i$, where $\hat{b}_i$ are the rows of $\hat{B}\hat{D}$ (see e.g. [12, p. 384] since weak Popov form implies row reduced). Thus, not all the $\hat{b}_i$ can have degree greater than $\frac{1}{\ell+1} \deg \det(B\hat{D})$, and so this bounds the degree of $\nu\hat{D}$.

Clearly $\det \hat{B} = \det \hat{M}$. Since $\hat{M}$ is an upper-diagonal matrix, we can therefore easily compute its determinant:

$$\deg \det \hat{M} = \deg \left( G^s(\ell-s+1) \prod_{i=1}^{s-1} G^i \right) = (s\ell - \binom{s}{2})n$$

Also $\deg \det \hat{D} = \mu_0 + \sum_{i} \eta_i = \binom{\ell+1}{2}(k-1) + 1$. Thus we have

$$\deg \hat{\lambda}_0 + \mu_0 \leq \deg(\nu\hat{D}) \leq \frac{1}{\ell+1} \left( (s\ell - \binom{s}{2})n + \binom{\ell+1}{2}(k-1) + 1 \right),$$

which rewrites into the sought bound.

When solving the key equations, we will seek a minimal degree vector in $\text{Row}_{\mathbb{F}[x]}(\hat{M}\hat{D})$ which has leading position 1. We are then hoping that the first element of this vector equals $\Lambda^s$. Since the minimal degree vector overall in the row space might not have leading position 1, the above corollary doesn’t quite state that we will surely fail in decoding when $\epsilon > \tau_{\text{row}}(s, \ell)$ However, it is natural to suspect that most likely, the minimal degree vector with leading position 1 has degree quite close to the minimal degree overall. Therefore, we might expect to fail. This intuition is also backed by simulation, see Section VII.

We can relate $\tau_{\text{row}}(s, \ell)$ to something very well known:

**Corollary 13:** Denote the maximal decoding radius of the Guruswami–Sudan algorithm on $C$ with multiplicity $s$ and list size $\ell$ by:

$$\tau_{\text{GS}}(s, \ell) = \frac{2\ell - s + 1}{2(\ell + 1)} n - \frac{\ell}{2s} (k-1) = \tau_{\text{row}}(s, \ell) + \frac{\ell}{s(\ell + 1)}$$

(see e.g. [21] Lemma 9.5]).

Taken over all $s$ and $\ell$, the decoding radius of Guruswami–Sudan describes a curve $J(n, d) = n - \sqrt{n(n-d)}$, often called the Johnson radius. For any integer $\tau < J(n, d)$ there exists infinitely many choices of $s$, $\ell$ such that $\tau = \lfloor \tau_{\text{GS}}(s, \ell) \rfloor$. Thus, by Corollary 13 Power decoding is similarly bounded by the Johnson radius. The corollary even tells us more: if we choose exactly the same $s$ and $\ell$ as in the Guruswami–Sudan, then Power decoding will decode up to the same radius or at most 1 less.

For the Guruswami–Sudan, good, closed-form expressions for small $s$ and $\ell$ given the code and $\tau$ can be found in [17] p. 53. These therefore immediately apply to Power decoding as well.

**VI. Failure Behaviour**

We will move on to investigate how Power decoding fails when at most $\tau_{\text{row}}(s, \ell)$ errors occur. There are two ways in which Algorithm 1 can give an unwanted answer: firstly, the algorithm can return fail; or secondly, the algorithm can return a different codeword than the sent one. For a specific sent codeword $c$ and received word $r$, we say that Power decoding fails if one of the two following conditions are satisfied:

1) Algorithm 1 returns fail.
2) There exists $c' \in C$, $c' \neq c$, and such that $\text{dist}(r, c') \leq \text{dist}(r, c)$.

Recall that when Algorithm 1 does not return fail, it always returns a codeword of minimal distance to the received. So if neither of the above conditions are satisfied, Algorithm 1 returns the correct answer. Contrarily, if only item 2 above is satisfied and $\text{dist}(r, c') = \text{dist}(r, c)$, then $c$ might still be correctly returned. This, however, depends rather arbitrarily on exactly which matrix $\hat{B}$ is computed by the weak Popov form algorithm. For the sake of a cleaner definition, we therefore consider this possibility as a failure as well.

We will begin with showing that the error vector alone determines whether the method succeeds. This drastically simplifying further examinations on the failure behaviour. It allows us first to show the—quite expected—property that the method never fails when fewer than $d/2$ errors occur. Secondly, it allows us to give a closed upper bound on the failure probability when $(s, \ell) = (2, 3)$.

**Proposition 14:** Power decoding succeeds for some received word $r$ if and only if it succeeds for $r + \hat{c}$ where $\hat{c}$ is any codeword.

**Proof:** If Power decoding fails for $r$ as received word, this is because there exist $\lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell \in \mathbb{F}[x]$ which solve Problem 3 and where $\lambda_0 \neq \Lambda^s$ and $\deg \lambda_0 \leq \deg \Lambda^s$. Assume this is the case. Let $\hat{R}$ be the Lagrange interpolant corresponding to $r + \hat{c}$ as received word, i.e. $\hat{R} = R + f$ where $f = ev^{-1}(\hat{c})$ and $\deg f < k$. We will show that there exist
\( \hat{\psi}_1, \ldots, \hat{\psi}_\ell \in \mathbb{F}[x] \) such that the \( \lambda_i, \hat{\psi}_\ell \) form a solution to Problem 3 for \(  \hat{R} \) in place of \( R \). Therefore, Power decoding will also fail for \( r + \hat{c} \) as received word.

Consider for \( t = 1, \ldots, \ell \) the following expansion:

\[
\sum_{i=0}^{\min(t,s-1)} \lambda_i \cdot \left( \binom{t}{i} \hat{R}^{t-i} G^i \right)
= \sum_{i=0}^{\min(t,s-1)} \lambda_i \left( \binom{t}{i} \sum_{h=0}^{t-i} \left( \binom{t-i}{h} \hat{R}^{t-i-h} f^h \right) G^i \right)
= \sum_{h=0}^{t} \hat{f}^h \sum_{i=0}^{\min(t-h,s-1)} \lambda_i \left( \binom{t}{i} \left( \binom{t-h}{i} \hat{R}^{t-h} G^i \right) \right)
\]

Note now that \( \binom{t}{i} \binom{n-s}{n-t} = \binom{t}{i} \binom{t-h}{i} \). Therefore, the above equals

\[
\sum_{h=0}^{t} \hat{f}^h \sum_{i=0}^{\min(t-h,s-1)} \lambda_i \left( \binom{t}{i} \left( \binom{t-h}{i} \hat{R}^{t-h} G^i \right) \right)
= \sum_{h=0}^{t} \hat{f}^h \psi_{t-h},
\]

where we by “\( \equiv \)” mean when \( t < s \) and congruent modulo \( G^s \) when \( t \geq s \). We set \( \hat{\psi}_\ell \) as the last expression above. By hypothesis, \( \deg \hat{\psi}_{t-h} - (t-h)(k-1) < \deg \lambda_0 \). Since \( \deg \hat{f} < k \) we therefore get \( \hat{\psi}_t - t(k-1) < \deg \lambda \).

This means the \( \lambda_i, \hat{\psi}_\ell \) indeed form a solution to Problem 3 for \(  \hat{R} \), as we set out to prove.

The proved implication can immediately be applied in the other direction since \( -\hat{c} \) is a codeword, showing the bi-implication.

We now prove that Power decoding always succeeds in half-the-minimum distance decoding. The proof is structured in a surprisingly complicated manner since we need to keep a handle on the key equations simultaneously.

**Proposition 15:** If fewer than \( d/2 \) errors occur, then Power decoding succeeds.

**Proof:** By Proposition 14 we can assume that \( 0 \) was sent. By Lemma 1 we then have \( R = -\Omega \Upsilon, \) where \( \Upsilon = G/\Lambda \).

Assume contrary to the proposition that Power decoding has failed. That means there exists (\( \lambda_0, \ldots, \lambda_{s-1}, \psi_1, \ldots, \psi_\ell \)) which solve Problem 3 and where \( \lambda_0 \neq \Lambda^s \) and \( \deg \lambda_0 \leq \deg \Lambda^s \). We will inductively establish \( P(t) \) for \( t = 0, \ldots, s - 1 \), where \( P(t) \) is the assertion

\[
P(t) : \quad \Lambda^{i+1} \mid \lambda_i \text{ and } \psi_{s-i} = 0 \text{ for } i = 0, \ldots, t .
\]

For \( t = s - 1 \), \( P(t) \) implies \( \Lambda^s \mid \psi_0 \), which contradicts the minimality of \( \lambda_0 \), finishing the proof.

For the case \( P(0) \), we need to prove that \( \Lambda \mid \lambda_0 \) and \( \psi_s = 0 \). Consider the \( s \)th key equation of Problem 3 which is satisfied by the \( \lambda_i \) and \( \psi_s \):

\[
\psi_s = \sum_{i=0}^{s-1} \binom{s}{i} \lambda_i R^{s-i} G^i \mod G^s
\]

(10)

\( \Upsilon^s \) divides each term of the summand, as well as the modulus \( G^s \), and so it must divide \( \psi_s \). However, we have

\[
\deg \psi_s \leq \deg \lambda_0 + s(k-1) \leq s \epsilon + s(k-1) < s(n - \epsilon) ,
\]

where the last inequality holds since \( 2s < n-k+1 \). Thus \( \psi_s = 0 \).

Returning to (10), we can then conclude \( \Lambda \mid \lambda_0 R^s \), since \( \Lambda \) divides every other term in the sum as well as the modulus. This implies \( \Lambda \mid \psi_0 \) which implies \( \gcd(\Lambda, R) = 1 \).

For the inductive step, assuming \( P(t - 1) \) we will prove \( P(t) \) for \( 1 \leq t < s \). Consider now the \( (s-t) \)th key equation, i.e.

\[
\psi_{s-t} = \sum_{i=0}^{s-t} \binom{s-t}{i} \lambda_i R^{s-t-i} G^i
\]

Similar to before, \( \Upsilon^{s-t} \) divides every term of the sum, so it divides \( \psi_{s-t} \). By \( P(t - 1) \) then \( \Lambda^{t-i} \mid \lambda_i \) for \( i = 0, \ldots, t-1 \), and therefore \( \Lambda^t \mid \psi_{s-t} \) and hence \( \Upsilon^{s-t} \Lambda^t \mid \psi_{s-t} \). But now we have

\[
\deg \psi_{s-t} \leq \deg \lambda_0 + (s-t)(k-1) \leq s \epsilon + (s-t)(k-1) < (s-t)(n - \epsilon) + t \epsilon ,
\]

which means \( \psi_{s-t} = 0 \).
It remains to show that $\Lambda^{s+1-i} | \lambda_i$ for $i = 0, \ldots, t - 1$. For $j = 1, \ldots, t$, multiply the $(s-j)$th key equation with $R^j$ and relax it to a congruence modulo $G^s$. We obtain $t + 1$ homogeneous linear equations in $\lambda_iR^{s-i}G^i$ of the form:

$$
0 \equiv \sum_{i=0}^{\min(s-1,s-j)} \binom{s-j}{i} (\lambda_iR^{s-i}G^i) \; \mod \; G^s, \quad j = 0, \ldots, t
$$

Subtracting the $j$th equation from the $(j-1)$st for $j = 1, \ldots, t$, we eliminate $\lambda_0$ and get

$$
0 \equiv \sum_{i=1}^{s-1} \binom{s-j}{i-1} (\lambda_iR^{s-i}G^i) \; \mod \; G^s, \quad j = 1, \ldots, t.
$$

This can be continued to get a series of equation systems, that is, for $t' = 1, \ldots, t$, we have a system:

$$
0 \equiv \sum_{i=t'}^{s-1} \binom{s-j}{i-t'} (\lambda_iR^{s-i}G^i) \; \mod \; G^s, \quad j = t', \ldots, t.
$$

For $t' = t$, the system (which is one equation) implies that $\Lambda^{t+1} | \lambda_t R^{s-t}G^s$ since $\Lambda^{t+1}$ divides all the sum's other terms and the modulus, and this implies $\Lambda | \lambda_t$. We can now go to the $t' = t - 1$ system and regard any of the two equations, and we conclude similarly that $\Lambda^{t+1} | \lambda_{t-1} R^{s-t+1}G^{t-1}$ since $\Lambda^{t+1}$ now is seen to divide all other terms of the sum as well as the modulus. This implies $\Lambda^2 | \lambda_{t-1}$. Continuing with decreasing $t'$ we can iteratively conclude $\Lambda^{t+1-\epsilon} | \lambda_{t'}$.

This finishes the induction step, establishing $P(t)$ for $t = 0, \ldots, s - 1$. As mentioned, this implies a contradiction, finishing the proof.

We are now in a position to bound the probability that Power decoding fails if errors of a given weight are drawn uniformly at random, for the case $(s, \ell) = (2, 3)$. Both result and proof are unfortunately quite technical, but the crux of it is the bound’s exponential dependence on $\tau_{Pow}(2, 3) - \epsilon$.

**Proposition 16:** Let $q = \# \mathbb{F}_{\ell}$. Whenever $\epsilon < \tau_{Pow}(2, 3)$, the probability that Power decoding fails is upper bounded by

$$
\begin{cases}
q^{-8(\tau_{Pow}(2, 3) - \epsilon) - 2} \cdot (q/(q-1))^2 2^\gamma \sum_{t=0}^{\ell} \binom{\ell}{t} 4^{t-\gamma} & \text{when } \gamma \geq 0 \\
q^{-2\epsilon + (n-2(k-1)) + 1} \cdot (q/(q-1))^2 2^\gamma \sum_{t=0}^{\ell} \binom{\ell}{t} 4^{t-\gamma} & \text{when } \gamma < 0
\end{cases}
$$

where $\gamma = 5\epsilon - (3n - 4(k-1))$.

**Proof:** By Proposition 14 we need only consider the probability over the choice of error vector, and not over the choice of sent codeword. Fix now the number of errors $\epsilon$ and error positions $\mathcal{E}$, implying a specific $\Lambda$. For each choice of error-value $e$ with these error positions, there is a unique polynomial $E$ with $\deg E < n$ and $E(\alpha_i) = e_i$; if $r = e$, then $E = R$ using our earlier notation. We will call the error-value, or $E$, “bad” if for $E$ there exist $\psi_1, \psi_2$ solving Problem 3 and such that $\lambda_0 \neq \Lambda$ while $\deg \lambda_0 \leq \deg \Lambda$. Consequently, Power decoding fails only for bad error-values. Denote by $S_{\Lambda} \subset \mathbb{F}[x]$ the set of bad $E$. We will give an upper bound $N$ on the size of $S_{\Lambda}$ and so $N/(q - 1)^{\epsilon}$ bounds the probability that for the fixed error positions, Power decoding fails (since for each position, we have $q - 1$ choices of an error value). $N$ will turn out to be independent of the choice of $\Lambda$, and thus $N/(q - 1)^{\epsilon}$ is a bound on the probability that Power decoding fails for any error of weight $\epsilon$.

By assumption, the following equations are satisfied:

$$
\psi_1 = \lambda_0 E + \lambda_1 G \\
\psi_2 \equiv \lambda_0 E^2 + 2\lambda_1 EG \mod G^2 \\
\psi_3 \equiv \lambda_0 E^3 + 3\lambda_1 E^2 G \mod G^2
$$

Since $E(\alpha_i) = 0$ whenever $i \notin \mathcal{E}$, then $\Upsilon \mid E$ where $\Upsilon = G/\Lambda$. Thus the above implies $\Upsilon \mid \psi_1$ and $\Upsilon^2 \mid \psi_t$ for $t = 2, 3$. Furthermore, we can conclude that $q \equiv \gcd(\psi_t, \Lambda)$ is the same for all $t$, since $g = \gcd(\psi_t, \Lambda)$. The regular form of the above three equations allows eliminating $\lambda_0$ and obtain:

$$
\begin{align*}
\psi_2 - E\psi_1 &\equiv \lambda_1 EG \mod G^2 \\
\psi_3 - E\psi_2 &\equiv \lambda_1 E^2 G \mod G^2
\end{align*}
$$

From this we first note that $G \mid (\psi_2 - E\psi_1)$. We will use this fact momentarily. With the two above equations we continue to eliminate $\lambda_1$ and rewrite:

$$
\begin{align*}
\psi_3 - E\psi_2 - E(\psi_2 - E\psi_1) &\equiv 0 \mod G^2 \\
E^2\psi_1 - 2E\psi_2 + \psi_3 &\equiv 0 \mod G^2 \\
E^2\psi_1^2 - 2E\psi_1\psi_2 + \psi_1\psi_3 &\equiv 0 \mod G^2 \\
(E\psi_1 - \psi_2)^2 + \psi_1\psi_3 &\equiv \psi_2^2 \mod G^2
\end{align*}
$$
But we concluded just before that \( G \mid (E\psi_1 - \psi_2) \) so \( (E\psi_1 - \psi_2)^2 \equiv 0 \mod G^2 \). This leaves the simple relation
\[
\begin{align*}
\hat{\psi}_2^2 &\equiv \psi_1^3 \mod G^2 \\
\hat{\psi}_2^2 \mathcal{Y} &\equiv \hat{\psi}_1 \hat{\psi}_3 \mod \Lambda^2,
\end{align*}
\]
where \( \hat{\psi}_t \triangleq \psi_t / \Psi^{\alpha}(t) \), and is a polynomial by our earlier observations. Thus, whenever \( E \) is bad, there is a triple \((\hat{\psi}_1, \hat{\psi}_2, \hat{\psi}_3) \in \mathbb{F}[X] \) satisfying the above relation as well as
\[
\deg \hat{\psi}_1 \leq d_t \triangleq 2c + t(k - 1) - \min(2, t)(n - \epsilon) .
\]
We will count the number of such triples momentarily. However, to thusly bound the number of bad error values, we have to determine how many different \( E \) could have the same triple. Recall that determining \( E \) up to congruence modulo \( \Lambda \) suffices, since this determines the error values. However, by our previous observation we have
\[
E \psi_1 \equiv \psi_2 \mod G \quad \iff \quad E \psi_1 \equiv \psi_2 \mathcal{Y} \mod \Lambda \quad \iff \quad E \equiv (\psi_2 \mathcal{Y} / g)(\psi_1 / g)^{-1} \mod \Lambda / g
\]
This means that for a given triple \((\hat{\psi}_1)_t\), having \( \gcd(\hat{\psi}_1, \Lambda) = g \), there can be at most \( q^{\deg g} \) possible choices of \( E \).

To bound the number of bad error values \( N \) for this given \( \Lambda \), we will therefore perform a weighted count of all triples satisfying (11) and (12), where a triple is counted with weight \( q^{\deg g} \), where \( g \) is a divisor of \( \Lambda \) dividing all the \( \hat{\psi}_t \):
\[
N \leq \sum_{g \mid \Lambda} q^{\deg g} \# \left\{ (\hat{\psi}_t)_t \in \mathbb{F}[X]^3 \mid g \mid \hat{\psi}_t, \deg \hat{\psi}_t \leq d_t, \mathcal{Y} \hat{\psi}_2^2 \equiv \hat{\psi}_1 \hat{\psi}_3 \mod \Lambda^2 \right\} = \sum_{g \mid \Lambda} q^{\deg g} \# \left\{ (\hat{\psi}_t)_t \in \mathbb{F}[X]^3 \mid \deg \hat{\psi}_t \leq d_t - \deg g, \mathcal{Y} \hat{\psi}_2^2 \equiv \hat{\psi}_1 \hat{\psi}_3 \mod (\Lambda / g)^2 \right\}
\]
Let \( T_g \) be the set inside the last sum. We use Lemma 17 (see below) to upper bound \( \#T_g \), for any choice of \( g \): setting \( A = (\Lambda / g)^2 \), \( B = \mathcal{Y} \) and \( K_t = d_t - \deg g \) in that lemma, we get
\[
\#T_g \leq 2^{\gamma + 2^2 - 2 \deg g} q^{4k - (2n - 2(k - 1)) + 1 + \max(0, \gamma) - \deg g}
\]
where \( \gamma = 5\epsilon - (3n - 4(k - 1)) \). This is only 2 dependent on the degree of \( g \). For each choice of \( \deg g \), we can select \( g \) in \( \left( \frac{\epsilon}{\deg g} \right) \) ways since \( g \mid A \). For the case \( \gamma \geq 0 \), this gives us:
\[
N \leq q^{\epsilon + 8(\epsilon - \tau_{\text{pow}}(2, 3)) - 2^2 \gamma} \sum_{t=0}^\epsilon \left( \frac{\epsilon}{t} \right) 4^{t - \epsilon}
\]
since \( 8\epsilon - (5n - 6(k - 1) - 3) = 8(\epsilon - \tau_{\text{pow}}(2, 3)) \).

A similar expression is obtained for the case \( \gamma < 0 \). As previously described \( N/(q - 1)^\epsilon \) then becomes a bound on the probability of decoding failure.

\textbf{Lemma 17:} Let \( A, B \in \mathbb{F}[X] \) with \( \gcd(A, B) = 1 \), and \( K_1 < K_2 < K_3 \in \mathbb{Z}_+ \), as well as \( q = \#\mathbb{F} \). Let \( S \) denote the set of triples \((f_1, f_2, f_3) \in \mathbb{F}[X]^3 \) such that \( B f_2^2 \equiv f_1 f_3 \mod A \), while \( \deg f_1 \leq K_1 \) and \( f_2 \) is monic. Then
\[
\#S \leq 2^{K_1 + K_3} q^{K_2 + 1 + \max(0, \gamma)},
\]
where \( \gamma = \max(K_1, K_3, 2K_2 + \deg B) - \deg A \).

\textbf{Proof:} Consider first \( \gamma < 0 \) in which case \( B f_2^2 = f_1 f_3 \). We can choose \( f_2 \) in \( q^{K_2 - 1} \) ways. The prime divisors of \( B f_2^2 \) should then be distributed among \( f_1 \) and \( f_3 \), which can be done in \( 2^{K_1 + K_3} \) ways. Finally, the leading coefficient of \( f_1 \) can be chosen in \( q - 1 \) ways.

Consider now \( \gamma \geq 0 \). We choose again first \( f_2 \) in one of \( q^{K_2 - 1} \) ways. Then \( f_1 f_3 \) must be in the set \( \{ B f_2^2 + pA \mid p \in \mathbb{F}[X], \deg p \leq \gamma, \} \), having cardinality at most \( q^{\gamma + 1} \). For each of these choices of \( f_1 f_3 \), we can again choose \( f_1 \) and \( f_3 \) in at most \( (q - 1)2^{K_1 + K_3} \) ways.

The bound of Proposition 16 does not seem to be tight. For instance, for a \([32, 9]\) code over \( GF(32) \), the proposition gives a failure probability above 1 for \( \epsilon = 13 \) errors, but simulations indicate that decoding succeeds almost always (see next section). Already for 12 errors on the same code, the proposition bounds the failure probability at \( \approx 6.2 \cdot 10^{-10} \). Similarly, for a \([256, 63]\) code over \( GF(256) \), the proposition is only non-trivial for \( \epsilon < 109 \) while in simulations, decoding works almost always up to \( \tau_{\text{pow}}(2, 3) = 112 \). However, in an asymptotic and relative sense, the proposition is tight:

\textbf{Corollary 18:} When \( s = 2 \) and \( \ell = 3 \), then for any \( \delta > 0 \), with \( n \to \infty \) while keeping \( q/n, k/n \) and \( \epsilon/n \) constant, the probability that Power decoding fails goes to 0 when \( \epsilon/n < \tau_{\text{pow}}(2, 3)/n - \delta \).
The contribution by cost of decoding that a significant practical impact on the running time, especially for higher-rate codes. We will now show that the re-encoding \( k/n \) asymptotic analysis where step in the Guruswami–Sudan algorithm [14]. It is especially powerful when using different multiplicities at each point, such as failures stem from this.

Power decoding always fails. We performed another simulation for this code \( r = c + e \) for some randomly chosen \( e \) (though, of course, Proposition [14] implies that shifting by \( e \) makes no difference). We have limited the decoding radii used to being \( \tau = \tau_{Pow}(s, \ell) + \{-1, 0, 1\} \). \( N \) was either \( 10^5 \) or \( 10^6 \) for a given parameter set. The results are listed as Table [1].

As is evident, \( \tau_{Pow}(s, \ell) \) very clearly describes the number of errors we can rely on correcting: the probability of failing appears to decay exponentially with \( \tau_{Pow}(s, \ell) - \epsilon \), as we might expect if extrapolating from the bound of Proposition [16]. In fact, the failure probability is so low that it is difficult to observe failing cases for randomly selected errors.

The case having the highest failure rate is the very low-rate code \([22, 3]_{GF(2^3)}\). For such a low-rate code, \( \tau_{Pow}(s, \ell) \) is quite close to the minimum distance, and there is a significant probability that a random error will yield a received codeword which is closer to another received word. In this case, Power decoding always fails. We performed another simulation for this code with \( 10^4 \) random errors of weight exactly 14 and decoding using the Guruswami–Sudan list decoder. This simulation had a \( 1.23 \times 10^{-2} \) probability that another codeword was as close or closer to the sent codeword. Thus most of the Power decoding failures stem from this.

### VII. Simulation Results

The proposed decoding algorithm has been implemented in Sage v6.6 [28], and is available for download at [http://jsrn.dk/code-for-articles](http://jsrn.dk/code-for-articles). The implementation uses the punctured model described in Section [IV-A] and computes the weak Popov form using the Mulders–Storjohann algorithm [15]. The asymptotic complexity of the implementation is therefore \( O((f, s, n)^2) \).

To evaluate the failure probability, we have selected a range of code and decoding parameters and run the algorithm for a large number of random errors. More precisely, for each set of parameters, and each decoding radius \( \tau \), we have created \( N \) random errors of weight exactly \( \tau \) and attempted to decode a received word \( r = c + e \) for some randomly chosen \( e \) (though, of course, Proposition [14] implies that shifting by \( e \) makes no difference). We have limited the decoding radii used to being \( \tau_{Pow}(s, \ell) + \{-1, 0, 1\} \). \( N \) was either \( 10^5 \) or \( 10^6 \) for a given parameter set. The results are listed as Table [1].

As is evident, \( \tau_{Pow}(s, \ell) \) very clearly describes the number of errors we can rely on correcting: the probability of failing appears to decay exponentially with \( \tau_{Pow}(s, \ell) - \epsilon \), as we might expect if extrapolating from the bound of Proposition [16]. In fact, the failure probability is so low that it is difficult to observe failing cases for randomly selected errors.

The case having the highest failure rate is the very low-rate code \([22, 3]_{GF(2^3)}\). For such a low-rate code, \( \tau_{Pow}(s, \ell) \) is quite close to the minimum distance, and there is a significant probability that a random error will yield a received codeword which is closer to another received word. In this case, Power decoding always fails. We performed another simulation for this code with \( 10^4 \) random errors of weight exactly 14 and decoding using the Guruswami–Sudan list decoder. This simulation had a \( 1.23 \times 10^{-2} \) probability that another codeword was as close or closer to the sent codeword. Thus most of the Power decoding failures stem from this.

### VIII. Re-Encoding

“Re-Encoding” is a simple technique invented by Kötter and Vardy, originally for reducing the complexity of the interpolation step in the Guruswami–Sudan algorithm [13]. It is especially powerful when using different multiplicities at each point, such as in the Kötter–Vardy soft-decision decoding version of Guruswami–Sudan [13]. For the regular Guruswami–Sudan, and in usual asymptotic analysis where \( k/n \) is considered a constant, re-encoding does not change the asymptotic cost; however, it can have a significant practical impact on the running time, especially for higher-rate codes. We will now show that the re-encoding transformation easily applies to Power decoding as well.

Consider that \( \hat{r} \) is the received word. Using Lagrange interpolation, we can easily compute the unique \( \hat{c} = ev(f) \in C \) such that \( \hat{c} \) and \( \hat{r} \) coincide on the first \( k \) positions. Clearly, decoding \( r = \hat{r} - \hat{c} \) immediately gives a decoding of \( \hat{r} \). The idea of re-encoding is that the leading \( k \) zeroes of the resulting \( r \) might be utilised in the decoding procedure to reduce the computation cost of decoding \( r \).

Assume therefore for this section that \( r \) is the received word after re-encoding and therefore has \( k \) leading zeroes. That means \( G \mid R \) where \( G = \prod_{i=1}^{k} (x - \alpha_i) \). Consider the linearised key equations of Problem [3]. Each of them are now divisible by \( G_{\min}(s, \ell) \), and so we obtain:

<table>
<thead>
<tr>
<th>([n, k]_F)</th>
<th>((s, \ell))</th>
<th>(\tau_{Pow})</th>
<th>(P_f(\tau_{Pow} - 1))</th>
<th>(P_f(\tau_{Pow}))</th>
<th>(P_f(\tau_{Pow} + 1))</th>
<th>(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[24, 7]_{GF(25)}</td>
<td>(2, 4)</td>
<td>10</td>
<td>0</td>
<td>6.800 \times 10^{-5}</td>
<td>1 - 5.8 \cdot 10^{-5}</td>
<td>10^5</td>
</tr>
<tr>
<td>[32, 9]_{GF(32)}</td>
<td>(2, 3)</td>
<td>13.5</td>
<td>0</td>
<td>0</td>
<td>1 - 4.2 \cdot 10^{-4}</td>
<td>10^6</td>
</tr>
<tr>
<td>[22, 3]_{GF(23)}</td>
<td>(6, 18)</td>
<td>14</td>
<td>3.45 \times 10^{-4}</td>
<td>1.214 \times 10^{-2}</td>
<td>1</td>
<td>10^6</td>
</tr>
<tr>
<td>[64, 29]_{GF(64)}</td>
<td>(4, 5)</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10^5</td>
</tr>
<tr>
<td>[68, 31]_{GF(71)}</td>
<td>(3, 4)</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10^6</td>
</tr>
<tr>
<td>[125, 51]_{GF(125)}</td>
<td>(4, 6)</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10^7</td>
</tr>
<tr>
<td>[256, 63]_{GF(256)}</td>
<td>(2, 4)</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>1 - 3.10 \times 10^{-4}</td>
<td>10^5</td>
</tr>
</tbody>
</table>

Table I

**Proof:** We will consider only the high-error failure probability of Proposition [16] the other case follows similarly. For \( n \to \infty \), the failure probability bound will approach

\[
q^{-\gamma(\tau_{Pow}(2,3)-\epsilon)} - 2^{\gamma} \sum_{t=0}^{\epsilon} \left( \frac{\epsilon}{t} \right) 4^{t} t! \leq q^{\gamma(\tau_{Pow}(2,3)-\epsilon)} - 2^{\gamma} \cdot 8^\epsilon \leq \left(q^n\right)^{\gamma(\tau_{Pow}(2,3)-\epsilon)/n - 2/n - (\gamma - 3\epsilon)/(n \log q - \log q^n)(\epsilon)}
\]

The contribution \( -2/n - (\gamma - 3\epsilon)/(n \log q - \log q^n)(\epsilon) \) goes to 0 as \( n, q \to \infty \), leaving \( q^n\) for \( a = -8(\tau_{Pow}(2,3)-\epsilon)/n = -8\delta \).
where the last inequality comes from the general assumption that 

\[ f \lambda^t \]

\[ \text{variants of the powered Lagrange interpolant} \]

\[ \text{A. Solving the Re-Encoded Equations} \]

Gentner–Augot–Zeh algorithm for interpolation in Guruswami–Sudan [33].

\[ \text{syndromes} \]

\[ \text{parameters would be slightly more complicated but not impossible.} \]

As is usual for syndrome-formulated key equations, we will assume that 

\[ \text{key equations of Theorem 2 can similarly be rewritten to be based on syndrome polynomials, which we will show in this} \]

\[ \text{[4], and similarly, Power decoding without multiplicities was first described using a similar list of key equations [25]. The} \]

\[ \text{obtaining} \]

\[ \text{M} \]

\[ \text{We then compute} \]

\[ \text{To find a minimal} \]

\[ \text{M} \]

\[ \text{We solve these key equations exactly in the same way as before: construct an} \]

\[ \text{Note that re-encoding will not change the failure behaviour: by Proposition 14, the re-encoded equations before dividing} \]

\[ \text{this row space with} \]

\[ \text{If} \]

\[ \text{Recall the reversal operator} \]

\[ \text{Furthermore, due to a non-essential technicality, we will assume} \]

\[ \text{Recall the reversal operator} [d]p \]

\[ \text{IX. Syndrome Key Equations} \]

As described in Section [1], the first key equation decoding algorithm was based on the notion of syndrome polynomial [4], and similarly, Power decoding without multiplicities was first described using a similar list of key equations [25]. The key equations of Theorem [2] can similarly be rewritten to be based on syndrome polynomials, which we will show in this section. As is usual for syndrome-formulated key equations, we will assume that 0 is not used as an evaluation point. Therefore \( x \nmid G \). Furthermore, due to a non-essential technicality, we will assume \( s < n \). If this did not hold, the following analysis of parameters would be slightly more complicated but not impossible.

Recall the reversal operator \([d]p \) which we defined in Section [II-B]. Define for a given value of the multiplicity \( s \) the following variants of the powered Lagrange interpolant \( R \) as well as a generalised notion of syndrome:

\[ R^{(i,t)} = R^{t-i} \mod G^{s-i} \]

\[ S^{(i,t)} = \frac{R^{(i,t)}}{G^{s-i}} \]

Note the degree that the reversal-operator on \( R^{(i,t)} \) uses: if \( t - i \leq s - i \) then \( R^{(i,t)} = R^{t-i} \) so the degree upper bound is \((t-i)(n-1)\). If \( t - i > s - i \) then \( \deg R^{t-i} > \deg G^{s-i} \) since we have assumed \( s < n \), and therefore \( \deg R^{(i,t)} \leq (s-i)n-1 \).

If \( s = 1 \) then \( S^{(1,t)} \) equals the classical syndrome polynomial \( S \) which we used in Section [II-B] and \( S^{(1,t)} \) equals the syndromes \( S^{(t)} \) discussed in Section [II-C]. A notion of generalised syndromes very related to \( S^{(t)} \) is also used in the Gentner–Augot–Zeh algorithm for interpolation in Guruswami–Sudan [33].
Theorem 19: For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then there exist $g_t \in \mathbb{F}[x]$ for $t = s, \ldots, \ell$ such that

$$
\sum_{i=0}^{t} \Lambda^{s-i} \Omega^i \binom{t}{i} S^{(i,t)} = 0 \mod x^0 \quad \text{for } t = 1, \ldots, s - 1
$$

$$
\sum_{i=0}^{t} \Lambda^{s-i} \Omega^i \binom{t}{i} S^{(i,t)} x^{t+i} = g_t \mod x^0 \quad \text{for } t = s, \ldots, \ell ,
$$

where

$$
\deg g_t \leq \begin{cases} 
\epsilon s - s & \text{if } t = s \\
\epsilon s - 1 & \text{if } t > s 
\end{cases}
$$

$$
\varrho_t = \begin{cases} 
t(n-k) & \text{if } t \leq s \\
sn - t(k - 1) - 1 & \text{otherwise} 
\end{cases}
$$

$$
t_{i,t} = \begin{cases} 
0 & \text{if } t = s \\
i & \text{if } t > s .
\end{cases}
$$

Proof: We need to distinguish between two cases: $t < s$ and $t \geq s$. Assume first $t < s$. Since $R^{(i,t)} = R^{t-i}$, Theorem 2 gives us

$$
\sum_{t=0}^{t} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i = \Lambda^s f^t \iff \sum_{t=0}^{t+\epsilon(n-1)} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i = \Lambda^s f^t ,
$$

where $\epsilon s + t(n - 1)$ arise from counting the degree upper bound on the left-hand side. Every term in the sum has the same degree bound, so we get

$$
\sum_{t=0}^{t} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i \equiv \Lambda^s f^t x^{t(n - k)} \iff \sum_{t=0}^{t} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i \equiv 0 \mod x^t x^{t(n - k)}
$$

$$
\sum_{t=0}^{t} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} S^{(i,t)} \equiv 0 \mod x^t x^{t(n - k)} ,
$$

where the last line follows from $G^i$ being invertible modulo $x^{t(n - k)}$. This concludes the case $t < s$.

For the case $t \geq s$, we proceed similarly. In the congruence of Theorem 2, we can readily replace $R^{t-i} G^i$ with $R^{(i,t)} G^i$ modulo $G^s$. This gives us:

$$
\Lambda^s f^t = \sum_{i=0}^{s-1} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i \mod G^s \iff \Lambda^s f^t + \overline{g}_t G^s = \sum_{i=0}^{s-1} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i ,
$$

for some $\overline{g}_t \in \mathbb{F}[x]$. The degree of the right-hand side is bounded as

$$
\max i \left\{ (s - i) \epsilon + i(\epsilon - 1) + \deg R^{(i,t)} + \deg x^t \right\} \leq \begin{cases} 
\epsilon s + s(n - 1 - \epsilon) & \text{if } t = s \\
\epsilon s + sn - 1 & \text{if } t > s 
\end{cases}
$$

This immediately bounds $\deg g_t$ as the theorem states. Note that the above equals $\varrho_t + s \epsilon + t(k - 1)$ in all cases. We can now reverse the equation as in the previous case. When $t > s$ then the degree bound on the summands are not all the same, so we must add powers of $x$ in the reversed expression:

$$
\Lambda^s f^t x^{t+i} + g_t G^s = \sum_{i=0}^{s-1} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i x^{t+i} \iff \overline{g}_t G^s = \sum_{i=0}^{s-1} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} R^{(i,t)} G^i x^{t+i} \mod x^t
$$

$$
g_t G^s \equiv \sum_{i=0}^{s-1} \left( \Lambda^{s-i} \Omega^i \right) \binom{t}{i} S^{(i,t)} x^{t+i} \mod x^t
$$
Remark 20: It follows from the derivation of Theorem 2 that, explicitly
\[ g_t = -\sum_{i=s}^{\ell} \binom{\ell}{i} \Lambda^{i-s} \Omega^{s-R^{t-i}}. \]
Looking at this explicit equation, the bound on \( g_t \) is surprising.

Theorem 19 leads to a decoding algorithm in the very same way as Theorem 2. We could call these algorithms “Power syndromes” and “Power Gao” respectively. We have the following important remark:

Corollary 21: Decoding using Power Gao succeeds if and only if decoding using Power syndromes succeeds.

Proof: This follows easily by the same transformation as in the proof of Theorem 19 a solution to the linearised key equations of Power Gao induces a solution to the linearised key equations of Power syndromes, and vice versa. ■

Thus the two decoding algorithms have exactly the same decoding performance.

A. Solving the Syndrome Equations

To use the key equations of Theorem 19 for decoding, we again proceed in a manner similar to that of Section IV we linearise the problem by forgetting the algebraic connection between the \( \Lambda^{s-i} \Omega^s \) and \( g_t \). The problem then becomes finding \( v = (\Lambda, \Lambda^{s-1} \Omega^s, \ldots, \Lambda \Omega^{s-1}, 0, \ldots, 0, g_s, \ldots, g_t) \) as the, hopefully, lowest weighted-degree vector in the row space of an explicit \( F[x] \)-matrix, \( M_{\text{Syn}} \), a problem we can solve by applying lattice basis reduction techniques. We will not go through all the details as in Section IV since the technique is so similar.

\[
M_{\text{Syn}} \in F[x]^{(s+\ell) \times (s+\ell)}
\]
becomes
\[
M_{\text{Syn}} = \begin{bmatrix}
I_{s \times s} & N_{\text{Syn}} \\
0_{s \times s} & \text{diag}(x^{\rho_1}, \ldots, x^{\rho_\ell})
\end{bmatrix},
\]
where \( N_{\text{Syn}} \in F[x]^{s \times s} \) is the matrix whose \((i, t)\)'th entry is
\[
N_{\text{Syn}}[i, t] = \binom{t}{i} \Lambda^{s-i} \Omega^s x^{i-t}, \quad i = 0, \ldots, s - 1 \text{ and } t = 1, \ldots, \ell.
\]
To find a low weighted-degree vector in the row space of \( M_{\text{Syn}} \), we again transform \( M_{\text{Syn}} D_{\text{Syn}} \) into weak Popov form, where \( D_{\text{Syn}} \) is an appropriately chosen diagonal matrix, containing only powers of \( x \). \( D_{\text{Syn}} \) should be chosen such that the degrees of the entries of the sought solution \( v D_{\text{Syn}} \) all have roughly the same degrees while retaining \( \text{LP}(v D_{\text{Syn}}) = 1 \). To handle the first 0 entries, we simply use a large weight. Explicitly, we can select:
\[
D_{\text{Syn}} = \text{diag}(x^1, x^1, x^2, \ldots, x^{s-1}, x^{s \tau_{\text{pow}}(s, \ell)}, \ldots, x^{s \tau_{\text{pow}}(s, \ell)}, x^s, x^1, \ldots, x^1)
\]
where \( \tau_{\text{pow}}(s, \ell) \) is the largest power of \( x \) that is less than or equal to \( s \). We then compute \( B_{\text{Syn}} \) unimodular equivalent to \( M_{\text{Syn}} \) and such that \( B_{\text{Syn}} D_{\text{Syn}} \) is in weak Popov form. If we are fortunate, \( v D_{\text{Syn}} \) will be the lowest-degree row in \( B_{\text{Syn}} D_{\text{Syn}} \) having leading position on the first position, up to an \( F \)-multiple. From this, we immediately get \( \Lambda \) and \( \Omega \), and we calculate \( f \) using e.g. Lemma 1.

Similarly to the re-encoding case in Section VIII the cost of computing \( B_{\text{Syn}} \) seems to be lower than minimising \( MD \) directly, but not asymptotically so. In particular, the asymptotic complexity has quasi-linear dependence on \( \deg(M_{\text{Syn}} D_{\text{Syn}}) \in O(s(n-k)) \). This is similar to when using the re-encoding technique of Section VIII. However, the puncturing of the module done in Section IV-A cannot be replicated for Power syndrome since that would result in an over-defined basis of the resulting punctured module (the punctured matrix would have more rows than columns). This in turn would drastically reduce the decoding radius. Thus, the Power syndrome variant must reduce an \((s+\ell)^2\) matrix, while Power Gao, with or without re-encoding, reduces an \((\ell+1)^2\) matrix. On the other hand, it might be possible that the \( s - 1 \) 0-remainder congruences of Power syndrome could be handled in a faster manner than here described. Without a much finer analysis and concrete choices of algorithms for computing the weak Popov form, we cannot conclude which algorithm will be fastest and by how much.

X. CONCLUSION

We demonstrated how the Power decoding technique for Reed–Solomon codes can be augmented with a new parameter—the multiplicity—to attain the Johnson decoding radius. The resulting decoder is, as the original Power decoding algorithm of Schmidt, Sidorenko and Bossert, a partial decoder which has a low but non-zero probability of failing. The main advantage over the list decoding algorithms of Guruswami and Sudan or Wu is that one does not need \( F[x][y] \) root-finding.

We showed how one can efficiently solve the resulting key equations using lattice basis reduction techniques to obtain a complexity close to the fastest realisation of the Guruswami–Sudan or Wu list decoding algorithms: \( O((\ell^2) s n) \).

The exact failure behaviour of the decoding method remains largely open. For \( s = 1 \), i.e. the original Power decoding, the failure probability has previously been bounded only for \( \ell = 2, 3 \). The case \( s > 1 \) seems no easier to analyse. Proposition 14
simplifies the equations one needs to analyse, and this was instrumental in the case for which we were able to bound the failure probability: \((s, \ell) = (2, 3)\). These open questions on the failure probability were counterbalanced by simulation results on a range of code parameters which demonstrates a failure probability which decays exponentially fast as the number of errors is reduced.

We also discussed two variants of the decoding method which reduces the cost in practice: re-encoding and a syndrome formulation. Either method roughly replaces the complexity dependency on \(n\) with \(n - k\). More detailed analysis, and concrete choices of basis reduction algorithms is necessary to determine which one is fastest in practice.

**XI. Acknowledgements**

The author would like to thank Vladimir Sidorenko, Martin Bossert and Daniel Augot for discussions on Power decoding. The author gratefully acknowledges the support of the Digiteo foundation, project [IdealCodes], and also, while the author was with Ulm University, the support of the German Research Council “Deutsche Forschungsgemeinschaft” (DFG) under grant BO 867/22-1.

**References**


