Power decoding Reed-Solomon codes up to the Johnson radius

Rosenkilde, Johan Sebastian Heesemann

Published in:
Advances in Mathematics of Communications

Link to article, DOI:
10.3934/amc.2018005

Publication date:
2018

Document Version
Early version, also known as pre-print

Citation (APA):
Rosenkilde, J. S. H. (2018). Power decoding Reed-Solomon codes up to the Johnson radius. *Advances in Mathematics of Communications*, *12*(1), 81-106. https://doi.org/10.3934/amc.2018005

Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **20xx**

doi:10.3934/amc.xx.xx.xx

pp. **X–XX**

POWER DECODING REED–SOLOMON CODES UP TO THE JOHNSON RADIUS

Johan Rosenkilde

Technical University of Denmark,
Department of Applied Mathematics and Computer Science
Denmark

(Communicated by the associate editor name)

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 is possible to use this approach to decode up to the Johnson radius – the decoding radius of the Guruswami–Sudan algorithm. In this paper we show that this can be done by incorporating a notion of multiplicities. As the original Power decoding, the proposed algorithm is a one-pass algorithm: decoding follows immediately from solving a shift-register type equation, which we show can be done in quasi-linear time. It is a "partial bounded-distance decoding algorithm" since it will fail to return a codeword for a few error patterns within its decoding radius; we investigate its failure behaviour theoretically as well as give simulation results. This is an extended version where we also show how the method can be made practically faster using a reencoding or a syndrome formulation.

1. **Introduction.**

Power decoding was originally proposed by Schmidt, Sidorenko and Bossert for low-rate Reed–Solomon codes (RS) [41]. Using shift-register synthesis techniques, the method can decode as many errors as the Sudan algorithm [47]. As opposed to Sudan's list decoder, Power decoding is a one-pass algorithm where decoding is realised by solving a simultaneous shift-register problem; however, Power decoding always returns at most one codeword and will for a few error patterns simply fail. Simulations indicate that this occurs very rarely for random errors[1]

The Sudan decoder generalises to the Guruswami–Sudan decoder [17] by introducing the multiplicity parameter, improving the decoding radius for all rates up to the Johnson radius [18]. Since [41], 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 radius.

We settle this question in the affirmative. The overall behaviour of the obtained 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.

2010 *Mathematics Subject Classification.* Primary: TODO TODO; Secondary: TODO.
Key words and phrases. TODO TODO TODO.

[1]This behaviour was described as "probabilistic decoding" in e.g. [41, 42]. However, that term is usually reserved for randomised algorithms such as Las Vegas probabilistic algorithms. Power decoding is entirely deterministic, given the input so we prefer the term "partial decoding".

2. The decoding radius becomes almost exactly that of the Guruswami–Sudan decoder (under the same choices of parameters).
3. When a codeword is returned, it is always a closest codeword. The method will fail for a few error patterns whenever one decodes beyond half the minimum distance.

Furthermore, we will show how to realise the decoder efficiently using existing algorithms for solving simultaneous Hermite Padé approximations: using the algorithms of [15, 16] the complexity becomes $O^{\sim}(\ell^{\omega} sn)$, where ω is the exponent of matrix multiplication, and s, ℓ are the multiplicity, respectively powering parameters of the decoder, and $O^{\sim}(\cdot)$ is big-O but ignoring $\log(ns\ell)$ factors. Note that we always have $\ell \geq s$. The slightly better complexity $O^{\sim}(\ell^{\omega-1} s^2 n)$ can be achieved by relying on [38] (not yet published). The latter matches the best known complexity for the Guruswami–Sudan algorithm or the Wu list decoder [10].

We also investigate the failure behaviour of the proposed method. Though we do not settle the question of precisely how often Power decoding fails, we make some headway: we prove that the behaviour depends only on the error, and not the sent codeword, and we show that failure can only occur beyond half the minimum distance. We then give a closed upper bound on the probability for one choice of the algorithm's parameters, $(s, \ell) = (2, 3)$. We also present simulation results that demonstrate a very small probability of failure for larger parameter choices.

Compared to the two existing list decoders, the Guruswami–Sudan and the Wu algorithm, we believe the proposed algorithm is interesting for several reasons. Firstly, it is more practical since the decoder needs only a single sub-algorithm—of shift-register type—while the other two decoders are more involved. The Guruswami–Sudan algorithm consists of two steps: interpolation and root-finding. Interpolation is comparable to the computation in Power decoding (see next section), but root-finding is an additional, non-trivial step. E.g. in [1], a hardware implementation of Köetter and Vardy's soft-decision variant of Guruswami–Sudan has the critical path in the root-finding unit, and this also uses a significant area of the entire circuit. In practical applications of hard-decision decoding, one would likely use smaller multiplicities than in [1]: this would leave the interpolation unit significantly simpler than that of [1] while the root-finding unit would be unchanged, resulting in root-finding occupying relatively even more area and latency.

Secondly, Power decoding and the Guruswami–Sudan algorithm can both be adapted to other algebraic constructions, but not always with the same pros and cons; see e.g. [24, 25, 29, 51] for adaptions of Power decoding. Case in point, the proposed extension of Power decoding of this paper has already been adapted to improved, quasi-linear time decoding of Interleaved RS codes [36]. The previous only known algorithm with the same decoding radius was [12] which can be seen as an adapted Guruswami–Sudan, and which has poor complexity exactly because of the root-finding step.

Lastly, Power decoding decodes beyond half-the-minimum distance without the use of interpolation at all. This is a rarity in algebraic decoding, and the present paper demonstrates that the Johnson radius can be reached purely linear-algebraically; at least in the presence of random errors. This sheds new light on decoding of RS codes, and represents an important puzzle piece in relation to the two list-decoding algorithms.

Parts of these results were presented at ACCT-14 [33].

1.1. **Related Work.** Power decoding was introduced in [41,42]: for low-rate RS codes, it was shown how one can compute generalised syndromes from "powering" the received word, and that these can be used for efficient decoding by solving a multi-sequence shift-register synthesis problem. One chooses a "powering degree" ℓ: higher ℓ yields better decoding radius, but is admissible only for lower-rate codes. In [42], a bound on the failure probability was given for RS codes over binary extension fields when $\ell = 2$, but a general conjecture was given based on simulations results. The failure behaviour was then further examined in [54] and [32], where bounds on the failure probability were obtained over any field for $\ell = 2$ and $\ell = 3$. In [32], a reformulation of Power decoding was given based on Gao's decoder [14], and this was used to show that whether or not Power decoding fails depends only on the error pattern, and not the sent codeword.

The Guruswami–Sudan algorithm [17] is a polynomial-time list-decoding algorithm up to the Johnson radius $J_{n,k} = n - \sqrt{n(k-1)}$ [18]. "List-decoding" means that the algorithm will return *all* codewords within the decoding radius. For the algorithm one chooses two parameters $s, \ell \in \mathbb{Z}_+$, usually dubbed "the multiplicity" respectively "the list size". They satisfy $s \leq \ell$, and they need to grow large for attaining the best decoding radius: for a decoding radius of $J_{n,k} - \varepsilon n$, one needs $s, \ell \in O(1/\varepsilon)$ for any $\varepsilon \in \mathbb{R}_+$. See [31, p. 58] for an extreme numerical example with $\varepsilon \approx 1/n^2$.

As noted already in [42], Power decoding is related to Guruswami–Sudan when $s = 1$ (also known as "Sudan decoding" after [47]): choosing the same value for ℓ yields (almost) exactly the same decoding radius. Computationally, there are more similarities, as noted below.

Guruswami–Sudan consists of two phases, usually dubbed "Interpolation" and "Root-finding": first, one finds an "interpolation polynomial" $Q(y) \in \mathbb{F}[x][y]$, and then one finds $\mathbb{F}[x]$-roots of it. Both phases have received a tremendous amount of attention with the aim of speeding them up, e.g. [2,6,10,11,22,40,53]; see [10] for an overview on the literature for the Interpolation step. The best currently known complexities are $O^{\sim}(\ell^{\omega-1}s^2n)$ for Interpolation [10], and $O^{\sim}(\ell sn)$ for Root finding [28], if $|\mathbb{F}| \in O(n)$. Without the use of fast arithmetic, the best known complexities are $O(\ell^3 s^2 n^2)$ for Interpolation [53], respectively $O(\ell^2 s^2 n^2)$ for Root finding [40].

One approach for fast Interpolation in Guruswami–Sudan has been to formulate "Interpolation key equations", as in [40] for the case $s = 1$, and [53] for the general case. These are shift-register-type equations whose solution result in an interpolation polynomial. These are related to Power decoding: the generalised syndromes in [40] equal those of the original Power decoding [41]. However, the two sets of key equations are inherently *different*: the solution to the Power decoding equations yields the error locator, while no clear notion of an error locator is known for the Guruswami–Sudan algorithm. Similarly, the key equations that we derive in Section 3 bears a resemblance to the equations of [53], and it is an interesting question what the algebraic relation between the two approaches is.

The Wu decoding algorithm [52] is an amalgamation between classical key equation decoding [8] and the Guruswami–Sudan: one first attempts half-the-minimum distance decoding using the classical key equation (see the following section). If this fails, the polynomials computed in the failed attempt are then used to set up a problem solvable by an $\mathbb{F}(x)$-variant of the Guruswami–Sudan algorithm. One again needs Interpolation and Root-finding sub-algorithms which are similar to, but slightly more involved than, for Guruswami–Sudan; see e.g. [7, 10, 49] for work on these. The best complexities for these steps equal those of the Guruswami–Sudan algorithm [7,10]. However, from a practical perspective, the Wu algorithm is slightly more complicated to implement. The Wu algorithm is also a list-decoding algorithm, and also decodes up to $J_{n,k}$. Also here one needs to choose parameters s, ℓ, whose growth relate to the decoding radius as in the Guruswami–Sudan algorithm [7].

1.2. **Organisation.** In Section 2 we give an introduction to the previous key equation-based decoding algorithms: half-the-minimum distance and Power decoding. In Section 3, we then derive the new key equations: non-linear relations between known polynomials, revealing the error. We derive a decoding radius in Section 4, and relate it directly to that of the Guruswami–Sudan algorithm. Power decoding will fail on certain error patterns within this radius, however, and we investigate this in Section 5. In Section 6 we give simulation results. In Section 7 we show how to efficiently solve the key equations. In Section 8 and Section 9 we investigate re-encoding respectively syndrome reformulations of the proposed key equations, providing practical – if not asymptotic – speedups to the decoder.

The decoding method has been implemented in Sage v8.0 [45] and can be downloaded from `http://jsrn.dk/code-for-articles`, together with the code for running the simulation.

2. **Preliminaries and Existing Key Equations.**

In complexity discussions, we count arithmetic operations in the field $\mathbb{F}$. We will use ω as the exponent for matrix multiplication, i.e. $2 \leq \omega \leq 3$. We use $O^{\sim}(\cdot)$ as big-O but ignoring log-factors. In a few places we also use $\mathsf{M}(n)$ to denote the complexity of multiplying together two polynomials of degree at most n; we can trivially use $\mathsf{M}(n) \in O(n^2)$ or we can have $\mathsf{M}(n) \in O^{\sim}(n)$, see e.g. [50].

2.1. **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

$$\mathrm{ev}(f) = \big(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)\big) \ .$$

The $[n, k, d]$ Generalised Reed-Solomon (GRS) code for these parameters is the set

$$\mathcal{C} = \big\{\mathrm{ev}(f) \mid f \in \mathbb{F}[x],\ \deg f < k\big\} \subseteq \mathbb{F}^n \ .$$

The α_i are called *evaluation points* and the β_i *column multipliers*. $\mathcal{C}$ has minimum distance $d = n - k + 1$ which is the maximal possible according to the Singleton bound.

Consider now that some $\boldsymbol{c} = (c_1, \ldots, c_n)$ was sent with $\boldsymbol{c} = \mathrm{ev}(f)$ for some $f \in \mathbb{F}[x]$, and that $\boldsymbol{r} = (r_1, \ldots, r_n) = \boldsymbol{c} + \boldsymbol{e}$ was the received word with error $\boldsymbol{e} = (e_1, \ldots, e_n)$. Let $\mathcal{E} = \{i \mid e_i \neq 0\}$ and $\epsilon = |\mathcal{E}|$.

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

Introduce two essential polynomials, immediately computable by the receiver:

$$G = \prod_{i=1}^{n}(x - \alpha_i) \qquad\qquad R : \deg R < n,\ R(\alpha_i) = r_i, \quad i = 1, \ldots, n \ .$$

G can be pre-computed, while R is computed upon receiving $\boldsymbol{r}$ using Lagrange interpolation.

Key equation decoders revolve around the notion of an error locator Λ and error evaluator Ω:

$$\Lambda = \prod_{j \in \mathcal{E}} (x - \alpha_j) \qquad \Omega = -\sum_{i \in \mathcal{E}} e_i \zeta_i \prod_{j \in \mathcal{E} \setminus \{i\}} (x - \alpha_j) \ .$$

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

The following simple relation is at the heart of our investigations:

Lemma 2.1. $\Lambda(f - R) = \Omega G$.

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 $\boldsymbol{c}, \boldsymbol{r}, \boldsymbol{e}, \Lambda$, etc. introduced here will be used in the remainder of the article.

2.2. Classical Key Equations.

Let us revisit the key equation implicit in Gao's decoder [14], which follows directly from Lemma 2.1:

$$\Lambda R \equiv \Lambda f \mod G \ . \tag{1}$$

This is a non-linear equation in the unknowns Λ and f, and it is not immediately obvious how to build an efficient decoder around it. The good - and classical - idea is to *linearise* the relation: we replace the sought quantities Λ and Λf with unknowns λ and ψ, both in $\mathbb{F}[x]$, and such that

$$\lambda R \equiv \psi \mod G \ .$$

This is now a linear relation with infinitely many solutions. We further restrict the solutions by requiring

$$\deg \lambda + k - 1 \geq \deg \psi \ .$$

Note that this is satisfied if λ is replaced by Λ and ψ by Λf. Finally, we seek such λ, ψ where λ 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 Λ. Solving such requirements for λ and ψ is sometimes known as rational function reconstruction [50] or Padé approximation [3]. They are easy to solve for in complexity $O(n^2)$ or $O^{\sim}(n)$, using e.g. the extended Euclidean algorithm [13, 14, 48].

It can be shown that whenever $\epsilon < d/2$ we get $\lambda = \Lambda$ and $\psi = \Lambda f$, see e.g. [14]. Then $f = \psi/\lambda$ and decoding is finished. However, whenever $\epsilon \geq d/2$, the approach will not work, i.e. the found λ will not equal Λ.

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* [8]. First some notation: for $p \in \mathbb{F}[x]$, let $\mathsf{rev}_d(p)$ denote the *reversal of the coefficients* of p at degree d, i.e. $\mathsf{rev}_d(p) = x^d p(x^{-1})$ for some integer $d \geq \deg p$. To lighten the notation, we will often omit the d-argument when there is an implied upper bound on the degree of the polynomial being reversed; to be precise, note that we then 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[2] of $\mathsf{rev}(R)/\mathsf{rev}(G)$ truncated at x^{n-k}. Then by reversing Lemma 2.1 at degree $\epsilon+n-1$ we get:

$$\begin{aligned} \Lambda R &= \Lambda f - \Omega G && \iff \\ \mathsf{rev}_{\epsilon+n-1}(\Lambda R) &= \mathsf{rev}_{\epsilon+k-1}(\Lambda f)x^{n-k} - \mathsf{rev}_{\epsilon+n-1}(\Omega G) && \implies \\ \mathsf{rev}(\Lambda)\mathsf{rev}(R) &\equiv -\mathsf{rev}(\Omega)\mathsf{rev}(G) \mod x^{n-k} \ . \end{aligned}$$

Since $x \nmid \mathsf{rev}(G)$ this implies the well-known formula:

$$\mathsf{rev}(\Lambda)S \equiv -\mathsf{rev}(\Omega) \mod x^{n-k} \ . \tag{2}$$

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

$$\begin{aligned} \hat{\lambda}S &\equiv \hat{\omega} \mod x^{n-k} \quad \text{and} \\ \deg\hat{\lambda} &> \deg\hat{\omega} \ . \end{aligned}$$

This time the modulus is a power of x; solving such an equation for $\hat{\lambda}$ and $\hat{\omega}$ is known as Padé approximations [3] or a linear feedback shift-register synthesis [39, Section 6.7]. It can be solved in complexity $O(n^2)$ or $O^{\sim}(n)$ 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 $\hat{\lambda} = \mathsf{rev}(\Lambda)$, whenever $\epsilon \leq \lfloor (d-1)/2 \rfloor$ [8]. Slightly stronger, one can show that the approach will succeed if and only if the Gao key equation approach succeeds [32].

2.3. **Simply Powered Key Equations.**

(Simple) Power decoding, or decoding by virtual interleaving [42], is a generalisation of (1) where not one but multiple non-linear relations between Λ and f are identified, essentially still based on Lemma 2.1. The original formulation of [42] is based on the classical syndrome key equation, while powering the Gao key equation was described in [32]. We will begin with the latter:

Lemma 2.2 (Simply Powered key equations). *For any $t \in \mathbb{Z}_+$ then*

$$\Lambda R^t \equiv \Lambda f^t \mod G \ .$$

Proof. By Lemma 2.1 we have

$$\Lambda f^t = \Lambda\big(R-(R-f)\big)^t = \Lambda R^t + \Lambda(R-f)(\dots) \equiv \Lambda R^t \mod G \ .$$

□

Again this gives non-linear relations between Λ and f. To solve them efficiently, we choose some ℓ and linearise the first ℓ of the equations, introducing unknowns $\lambda, \psi_1, \dots, \psi_\ell \in \mathbb{F}[x]$. We then solve for λ, ψ_t such that λ is monic and of minimal degree such that

$$\begin{aligned} \lambda R^t &\equiv \psi_t \mod G \ , \qquad t = 1, \dots, \ell \quad \text{and} \\ \deg\lambda &\geq \deg\psi_t - t(k-1) \ . \end{aligned}$$

Finally, we hope that the found $\lambda = \Lambda$. In that case $f = \psi_1/\lambda$ and decoding is finished.

[2]By 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. [39, Section 6.2].

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

$$\epsilon \leq \tfrac{\ell}{\ell+1} n - \tfrac{1}{2}\ell(k-1) - \tfrac{\ell}{\ell+1} \ . \tag{3}$$

This argument does not imply that we will necessarily succeed when the bound is satisfied: the constructed system might have spurious "false solutions" of degree less than or equal to that of Λ. In such rare cases decoding might fail for fewer errors than (3). Bounding the probability that this occurs has proven difficult: we now know upper bounds when $\ell = 2, 3$ [32, 42], and Schmidt, Sidorenko, and Bossert posed a conjecture, backed by simulation, on the probability in general [42].

From (3) we can determine the value of ℓ that maximise the decoding radius. Whenever $k/n > 1/3$, one should simply choose $\ell = 1$, i.e. classical key equation decoding. Thus simple Power decoding is only useful for low-rate codes. Note that (3) is almost the same bound as the Sudan decoding algorithm [47], which is the Guruswami–Sudan algorithm with multiplicity 1.

Power decoding was originally described using a syndrome formulation instead of (3) [41]: we restrict ourselves to the case where 0 is not an evaluation point, and we define $S^{(t)}$ as the power series expansion of $\mathsf{rev}(R^{(t)})/\mathsf{rev}(G)$ 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 it follows from Lemma 2.2, by the same rewriting as in Section 2.2 [32], that:

$$\mathsf{rev}(\Lambda) S^{(t)} \equiv -\mathsf{rev}(\Omega_t) \mod x^{n-t(k-1)-1}, \tag{4}$$

where Ω_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 [32].

For the Gao formulation, the computational problem is sometimes known as "vector rational function reconstruction" [35], and for the syndrome formulation as "simultaneous Padé approximation" [3] or "multi-sequence shift-register synthesis" [42]. Iterative algorithms with $O(\ell n^2)$ complexity can be found in [5, 30, 42, 44]. $O^{\sim}(\ell^{\omega} n)$ algorithms are in [9, 30, 43]. Recently, the improved complexity $O^{\sim}(\ell^{\omega-1} n)$ has been achieved [38].

3. **New Key Equations.**

In this section we describe the main result of the paper: a new generalisation of Power decoding where we introduce a second parameter, *the multiplicity*. The resulting relations will again be non-linear in Λ and f, and we will employ a linearisation strategy similar to before.

The generalised key equations are described in the following theorem:

Theorem 3.1. *For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then*

$$\Lambda^s f^t = \sum_{i=0}^{t} \left(\Lambda^{s-i}\Omega^i\right) \left(\binom{t}{i} R^{t-i} G^i\right) \qquad \text{for } t = 1, \ldots, s-1 \ ,$$

$$\Lambda^s f^t \equiv \sum_{i=0}^{s-1} \left(\Lambda^{s-i}\Omega^i\right) \left(\binom{t}{i} R^{t-i} G^i\right) \mod G^s \qquad \text{for } t = s, \ldots, \ell \ .$$

Proof. We simply rewrite

$$\begin{aligned}\Lambda^s f^t &= \Lambda^s (R + (f - R))^t \\ &= \sum_{i=0}^{t} \binom{t}{i} \Lambda^s (f-R)^i R^{t-i} \ .\end{aligned}$$

If $t < s$ then $\Lambda^s (f-R)^i = \Lambda^{s-i}\Omega^i G^i$ for each i by Lemma 2.1. This finishes the first part of the theorem.

If $t \geq s$ then for $i = s, \dots, \ell$, the summand equals $\binom{t}{i}\Lambda^{i-s}\Omega^s G^s R^{t-i}$ due to Lemma 2.1, 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 ℓ equations in the (algebraically related) "unknowns" $\Lambda^s, \Lambda^{s-1}\Omega, \dots, \Lambda\Omega^{s-1}$ as well as $\Lambda^s f, \dots, \Lambda^s f^\ell$. These are "key equations" in the following sense: inner products of the unknowns $\Lambda^{s-i}\Omega^i$ with vectors of known polynomials (the $\binom{t}{i} R^{t-i} G^i$) equal the unknowns $\Lambda^s f^t$ modulo G^s – and hence have surprisingly low degree.

The relations of Theorem 3.1 are highly non-linear and solving for Λ and f directly would be computationally infeasible. Instead we *linearise* the relations: derive weaker, linear relations from Theorem 3.1 which can be solved efficiently:

Problem 3.2. *Find a vector* $(\lambda_1, \dots \lambda_s, \psi_1, \dots, \psi_\ell) \in \mathbb{F}[x]^{s+\ell}$ *with* λ_1 *monic and such that the following requirements are satisfied:*

$$\begin{aligned}
&1a) \quad \psi_t = \sum_{i=0}^{t} \lambda_{i+1} \cdot \left(\binom{t}{i} R^{t-i} G^i\right) \ , && \textit{for } t = 1, \dots, s-1 \\
&1b) \quad \psi_t \equiv \sum_{i=0}^{s-1} \lambda_{i+1} \cdot \left(\binom{t}{i} R^{t-i} G^i\right) \mod G^s \ , && \textit{for } t = s, \dots, \ell \\
&2) \quad \deg \lambda_1 \geq \deg \lambda_{i+1} + i \ , && \textit{for } i = 1, \dots, s-1 \\
&3) \quad \deg \lambda_1 \geq \deg \psi_t - t(k-1) \ , && \textit{for } t = 1, \dots, \ell \ .
\end{aligned}$$

Clearly $\mathbf{\Lambda} = (\Lambda^s, \Lambda^{s-1}\Omega, \dots, \Lambda\Omega^{s-1}, \Lambda^s f, \dots, \Lambda^s f^\ell)$ satisfies the requirements. The strategy is to find a *minimal solution*, by which we mean that $\deg \lambda_1$ is minimal, and then hope that this solution is actually $\mathbf{\Lambda}$. If that turns out to be the case, decoding can be completed simply by computing $f = \psi_1/\lambda_1$. *Whether* we can expect that to be the case is addressed in Sections 4 and 5.

The complete decoding algorithm is given as Algorithm 1, where we assume a solver for Problem 3.2. Note that Problem 3.2 could be solved as a series of linear systems in the coefficients of the λ_i, one system for each guess at $\deg \lambda_1$. A much more efficient algorithm for solving Problem 3.2 is addressed in Section 7, where we obtain the complexity $O^\sim(s\ell^\omega n)$ for Algorithm 1 (or $O^\sim(s^2\ell^{\omega-1}n)$ relying on the unpublished [37]).

Remark 3.3. The shape of the equations of Theorem 3.1 bears a striking resemblance to certain approaches for solving the Interpolation phase in the Guruswami–Sudan algorithm: the $\mathbb{F}[x]$ module characterisation as in [6, 23], and the (intermediate) Interpolation key equations as in [53, Eqn. (31)]. However, the Guruswami–Sudan algorithm has, a priori, nothing to do with the error locator, and the true connection between the two sets of key equations is unclear. For instance, it is not known if one can easily obtain the error locator from a Guruswami–Sudan interpolation polynomial or vice versa.

Algorithm 1 Efficient Power Decoding with Multiplicities

```
Input: $\boldsymbol{r} \in \mathbb{F}^n$, $s, \ell \in \mathbb{Z}_+$.
Output: $\tilde{\boldsymbol{c}} \in \mathcal{C}$ such that $\operatorname{dist}(\tilde{\boldsymbol{c}}, \boldsymbol{r})$ is minimal among codewords in $\mathcal{C}$, or fail
1 $R \leftarrow$ the Lagrange interpolation polynomial such that $R(\alpha_i) = r_i, i = 1, \dots, n$.
2 Compute $\binom{t}{i} R^{t-i} G^i \bmod G^s$ for $i = 1, \dots, t$ and $t = 1, \dots, \ell$.
3 $(\lambda_1, \dots, \lambda_s, \psi_1, \dots, \psi_\ell) \leftarrow$ a solution to Problem 3.2 such that $\deg \lambda_1$ is minimal.
4 If $\lambda_1$ divides $\psi_1$, let $f \leftarrow \psi_1/\lambda_1$. Otherwise fail.
5 If $\operatorname{dist}(\boldsymbol{r}, \operatorname{ev}(f)) = \deg \lambda_1 / s$ then return $\operatorname{ev}(f)$. Otherwise fail.
```

Remark 3.4. The original Power decoding can be described by analogy with decoding of certain Interleaved RS codes [42]. It would be interesting to find a similar analogue for the key equations of Theorem 3.1.

4. Decoding Radius.

We will now discuss how many errors Algorithm 1 will usually be able to correct. 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 when the number of errors is at most this. In this section we will answer the first of these questions, and turn to the latter in Section 5.

The decoding radius upper bound that we will derive is based on linear algebra: when the number of errors ϵ is large enough, then solutions to Problem 3.2 that are smaller than the sought $\boldsymbol{\Lambda}$ will appear.

Proposition 4.1. *Consider a received word* $\boldsymbol{r}$ *and the corresponding instance of Problem 3.2. There is a vector* $\boldsymbol{v} = (\check{\lambda}_1, \dots, \check{\lambda}_s, \check{\psi}_1, \dots \check{\psi}_\ell)$ *satisfying Items 1a and 1b of Problem 3.2 as well as:*

$$\begin{aligned} &2') \quad s\tau_{\mathrm{Pow}}(s,\ell) \geq \deg \lambda_{i+1} + i \ , \qquad &&\textit{for } i = 0, \dots, s-1 \\ &3') \quad s\tau_{\mathrm{Pow}}(s,\ell) \geq \deg \psi_t - t(k-1) \ , \quad &&\textit{for } t = 1, \dots, \ell \ . \end{aligned}$$

where

$$\tau_{\mathrm{Pow}}(s,\ell) = \frac{2\ell - s + 1}{2(\ell+1)} n - \frac{\ell}{2s}(k-1) - \frac{\ell}{s(\ell+1)} \ . \tag{5}$$

If $\epsilon > \tau_{\mathrm{Pow}}(s,\ell)$ *then* $\deg \Lambda > \deg \check{\lambda}_1 / s$.

Proof. Satisfying Items 1a, 1b of Problem 3.2 as well as Items 2' and Items 3' above is a homogeneous linear set of restrictions in the coefficients of the λ_i: the linear combinations on the right-hand side of Items 1a and 1b should have bounded degree, either directly or reduced modulo G^s. If there are more coefficients than constraints, there will be a solution.

Let us write $\tau = \tau_{\mathrm{Pow}}(s,\ell)$ for brevity; we will derive that if τ satisfies (5), then there will be a solution to the homogeneous system. For every $t = 1, \dots, s-1$, Item 1a imposes C_t constraints, where:

$$\begin{aligned} C_t &= \deg(\mathrm{rhs}) - (s\tau + t(k-1)) \\ &= \max_{i=0,\dots,s-1} (s\tau - i + (n-1)(t-i) + in) - (s\tau + t(k-1)) \\ &= tn - t - t(k-1) = t(n-1-(k-1)) \ . \end{aligned}$$

For Item 1b, then ψ_t has bounded degree modulo G^s, so this gives for $t = s, \dots, \ell$:

$$C_t = sn - 1 - (s\tau + t(k-1))$$

We thus have a total number of constraints:

$$\sum_{t=1}^{\ell} C_t = \sum_{t=1}^{s-1} \big(t(n-1-(k-1))\big) + \sum_{t=s}^{\ell} \big(sn - s\tau - t(k-1) - 1\big)$$
$$= \tfrac{1}{2}(2\ell - s + 1)sn - \tbinom{\ell+1}{2}(k-1) - (\ell - s + 1)s\tau - (\tbinom{s}{2} + \ell - s + 1)$$

The total number of coefficients in $\lambda_1, \dots, \lambda_s$ is:

$$K = \sum_{i=0}^{s-1}(s\tau - i + 1) = s^2\tau - \tbinom{s}{2} + s$$

The condition for a guaranteed solution is then $K > \sum_{t=1,\dots,\ell} C_t$, i.e.:

$$(\ell+1)s\tau > \tfrac{1}{2}(2\ell - s + 1)sn - \tbinom{\ell+1}{2}(k-1) - \ell - 1 \ .$$

Thus, there must be a solution satisfying Items 1a, 1b, 2' and 3' for τ satisfying:

$$(\ell+1)s\tau = \tfrac{1}{2}(2\ell - s + 1)sn - \tbinom{\ell+1}{2}(k-1) - \ell \ .$$

□

The solution $\check\lambda_1, \dots, \check\lambda_s$ guaranteed by Proposition 4.1 will not necessarily solve Problem 3.2: it might e.g. be that $\deg \check\lambda_1 < \deg \check\lambda_2 + 1 \leq s\tau_{\mathrm{Pow}}(s, \ell)$. However, it is natural to suspect that, once there are solutions to the system of Proposition 4.1, there will be solutions with $\deg \check\lambda_1 = s\tau_{\mathrm{Pow}}(s, \ell)$, and such solutions will necessarily also solve Problem 3.2. The minimal solution to Problem 3.2 will in such cases not be $\mathbf{\Lambda}$ that we are looking for. Therefore, we might *expect* to fail, whenever $\epsilon > \tau_{\mathrm{Pow}}(s, \ell)$. This intuition is completely backed by simulation, see Section 6: with high probability, decoding seems to fail if $\epsilon > \tau_{\mathrm{Pow}}(s, \ell)$, but for a few error patterns it does succeed after all. We will therefore regard $\tau_{\mathrm{Pow}}(s, \ell)$ as the decoding radius of Algorithm 1.

The expression $\tau_{\mathrm{Pow}}(s, \ell)$ turns out to related to something very well known:

Corollary 4.2. *Denote the maximal decoding radius of the Guruswami–Sudan algorithm on $\mathcal{C}$ with multiplicity s and list size ℓ by $\tau_{\mathrm{GS}}(s, \ell)$. Then*

$$\tau_{\mathrm{GS}}(s, \ell) = \frac{2\ell - s + 1}{2(\ell+1)} n - \frac{\ell}{2s}(k-1) = \tau_{\mathrm{Pow}}(s, \ell) + \frac{\ell}{s(\ell+1)} \ .$$

(see e.g. [39, Lemma 9.5]).

Taken over all s and ℓ, the decoding radius of Guruswami–Sudan describes a curve $J(n, d) = n - \sqrt{n(n-d)}$, often called the Johnson radius after [18]. For any integer $\tau < J(n, d)$ there exists infinitely many choices of s, ℓ such that $\tau = \lfloor \tau_{\mathrm{GS}}(s, \ell) \rfloor$. Thus, by Corollary 4.2, Power decoding is similarly bounded by the Johnson radius (for $s, \ell \to \infty$ then $\tau_{\mathrm{GS}}(s, \ell) - \tau_{\mathrm{Pow}}(s, \ell) \to 0$). The corollary even allows us to use closed-form expressions for small s and ℓ already analysed for the Guruswami–Sudan algorithm:

Proposition 4.3. *Given a decoding radius $\tau < J(n, d) = n - \sqrt{n(n-d)}$, let $\tilde\tau := \tau + \frac{1}{s(\tau)}$. As long as $\tilde\tau < J(n, d)$ then $\tau_{\mathrm{Pow}}\big(s(\tilde\tau), \ell(\tilde\tau)\big) \geq \tau$, where*

$$s(\tau) = \lfloor s_{\min}(\tau) + 1 \rfloor \qquad\qquad \ell(\tau) = \left\lfloor \tfrac{n-\tau}{k-1} \cdot s(\tau) + \tfrac{1}{2} - \tfrac{\sqrt{D(\tau)}}{k-1} \right\rfloor \ ,$$

$$s_{\min}(\tau) = \frac{\tau(k-1)}{(n-\tau)^2 - n(k-1)}$$
$$D(\tau) = \big(s(\tau) - s_{\min}(\tau)\big) \cdot \big((n-\tau)^2 - n(k-1)\big) \cdot s(\tau) + \tfrac{(k-1)^2}{4}$$

Proof. Since $\tilde{\tau} < J(n,d)$, it is a valid decoding radius for the Guruswami–Sudan algorithm, and so by [31, p. 53], then $\tau_{\mathrm{GS}}\big(s(\tilde{\tau}), \ell(\tilde{\tau})\big) \geq \tilde{\tau}$. Therefore Corollary 4.2 gives us $\tau_{\mathrm{Pow}}(s,\ell) \geq \tau + \frac{1}{s(\tau)} - \frac{\ell(\tilde{\tau})}{s(\tilde{\tau})(\ell(\tilde{\tau})+1)}$, so we are done if $s(\tau) \leq s(\tilde{\tau})$. But $s_{\min}(\tau)$ is monotonically increasing for $0 < \tau < J(n,d)$ so $s(\tau)$ is non-decreasing, □

Remark 4.4. We remark that the condition of Proposition 4.3 that $\tilde{\tau} < J(n,d)$ seems almost always to be verified: for $n < 100$ an exhaustive search found only 50 choices of the triple (n,k,τ) for which it was not verified, and in 48 of these cases $n = k+3$. As an example of the tightness of the closed expressions, consider the large parameters $[n,k] = [243320, 131155]$: here the list size of Proposition 4.3 never exceeds the minimal possible by more than 1 for all possibly decoding radii.

5. Failure Behaviour.

We will move on to investigate how Power decoding fails when at most $\tau_{\mathrm{Pow}}(s,\ell)$ 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 $\boldsymbol{c}$ and received word $\boldsymbol{r}$, we say that Power decoding *fails* if one of the two following conditions are satisfied:

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

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 $\mathrm{dist}(\boldsymbol{r}, \boldsymbol{c}') = \mathrm{dist}(\boldsymbol{r}, \boldsymbol{c})$, then $\boldsymbol{c}$ might still be correctly returned. However, it is much more likely that the found solution to the key equation in Line 3 will be some mix of the solutions corresponding to the two errors $\boldsymbol{r} - \boldsymbol{c}$ and $\boldsymbol{r} - \boldsymbol{c}'$, in which case decoding will fail. 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 simplifies 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 $(s,\ell) = (2,3)$. Lastly, we discuss the relation between Power decoding failing and having multiple close codewords to the received word.

Proposition 5.1. *The success of Power decoding $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ depends only on the error $\boldsymbol{e}$.*

Proof. It suffices to show that if Power decoding fails for $\boldsymbol{r}$ as received word, then Power decoding also fails for $\boldsymbol{r} + \hat{\boldsymbol{c}}$ where $\hat{\boldsymbol{c}}$ is any codeword. If decoding fails on input $\boldsymbol{r}$ this is because there exist $\lambda_1, \ldots, \lambda_s, \psi_1, \ldots, \psi_\ell \in \mathbb{F}[x]$ which solve Problem 3.2, and where $\lambda_1 \neq \Lambda^s$ and $\deg \lambda_1 \leq \deg \Lambda^s$. Assume this is the case. Let $\hat{R}$ be the Lagrange interpolant corresponding to $\boldsymbol{r} + \hat{\boldsymbol{c}}$ as received word, i.e. $\hat{R} = R + \hat{f}$ where $\hat{f} = \mathrm{ev}^{-1}(\hat{\boldsymbol{c}})$ and $\deg \hat{f} < k$. We will show that there exist $\hat{\psi}_1, \ldots, \hat{\psi}_\ell \in \mathbb{F}[x]$ such that the $\lambda_i, \hat{\psi}_t$ form a solution to Problem 3.2 for $\hat{R}$ in place of R. Therefore, Power decoding will also fail for $\boldsymbol{r} + \hat{\boldsymbol{c}}$ as received word.

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

$$\begin{aligned}
&\sum_{i=0}^{\min(t,s-1)} \lambda_{i+1} \cdot \left(\binom{t}{i} \hat{R}^{t-i} G^i \right) \\
&= \sum_{i=0}^{\min(t,s-1)} \lambda_{i+1} \binom{t}{i} \left(\sum_{h=0}^{t-i} \binom{t-i}{h} R^{t-i-h} \hat{f}^h \right) G^i \\
&= \sum_{h=0}^{t} \hat{f}^h \sum_{i=0}^{\min(t-h,s-1)} \lambda_{i+1} \binom{t}{i} \binom{t-i}{h} R^{t-i-h} G^i \ .
\end{aligned}$$

Note now that $\binom{t}{i}\binom{t-i}{h} = \binom{t}{h}\binom{t-h}{i}$. Therefore, the above equals

$$\begin{aligned}
&\sum_{h=0}^{t} \binom{t}{h} \hat{f}^h \sum_{i=0}^{\min(t-h,s-1)} \lambda_{i+1} \binom{t-h}{i} R^{t-i-h} G^i \\
&\equiv \sum_{h=0}^{t} \binom{t}{h} \hat{f}^h \psi_{t-h} \ ,
\end{aligned}$$

where we by "$\equiv$" mean $=$ when $t < s$ and congruent modulo G^s when $t \geq s$. We set $\hat{\psi}_t$ as the last expression above. By hypothesis, $\deg \psi_{t-h} - (t-h)(k-1) < \deg \lambda_1$. Since $\deg \hat{f} < k$ we therefore get $\hat{\psi}_t - t(k-1) < \deg \lambda$.

This means the λ_i, ψ_t indeed form a solution to Problem 3.2 for $\hat{R}$, as we set out to prove.

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

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

Proposition 5.2. *If fewer than $d/2$ errors occur, then Power decoding succeeds.*

Proof. By Proposition 5.1, we can assume that $\mathbf{0}$ was sent. By Lemma 2.1 we then have $R = -\Omega\Upsilon$, where $\Upsilon = G/\Lambda$.

Assume contrary to the proposition that Power decoding has failed. That means there exists $(\lambda_1, \ldots, \lambda_s, \psi_1, \ldots, \psi_\ell)$ which solve Problem 3.2, and where $\lambda_1 \neq \Lambda^s$ and $\deg \lambda_1 \leq \deg \Lambda^s$. We will inductively establish $P(t)$ for $t = 0, \ldots, s-1$, where $P(t)$ is the assertion

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

For $t = s-1$, $P(t)$ implies $\Lambda^s \mid \lambda_1$, which contradicts the minimality of λ_1, finishing the proof.

For the case $P(0)$, we need to prove that $\Lambda \mid \lambda_1$ and $\psi_s = 0$. Consider the s'th key equation of Problem 3.2 which is satisfied by the λ_{i+1} and ψ_s:

$$\psi_s \equiv \sum_{i=0}^{s-1} \binom{s}{i} \lambda_{i+1} R^{s-i} G^i \mod G^s \ . \tag{6}$$

Υ^s divides each term of the summand, as well as the modulus G^s, and so it must divide ψ_s. However, we have

$$\deg \psi_s \leq \deg \lambda_1 + s(k-1) \leq s\epsilon + s(k-1) < s(n-\epsilon) \ ,$$

where the last inequality holds since $2\epsilon < n-k+1$. Thus $\psi_s = 0$.

Returning to (6), we can then conclude $\Lambda \mid \lambda_1 R^s$, since Λ divides every other term in the sum as well as the modulus. This implies $\Lambda \mid \lambda_1$ since $\gcd(\Lambda, R) = 1$.

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.

$$\psi_{s-t} = \sum_{i=0}^{s-t} \binom{s-t}{i} \lambda_{i+1} R^{s-t-i} G^i \ .$$

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

$$\deg \psi_{s-t} \leq \deg \lambda_1 + (s-t)(k-1) \leq s\epsilon + (s-t)(k-1) < (s-t)(n-\epsilon) + t\epsilon \ ,$$

which means $\psi_{s-t} = 0$.

It remains to show that $\Lambda^{t+1-i} \mid \lambda_{i+1}$ for $i = 0, \ldots, t$. 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 $\lambda_{i+1} R^{s-i} G^i$ of the form:

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

Subtracting the jth equation from the $(j-1)$st for $j = 1, \ldots, t$, we eliminate λ_1 and get

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

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

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

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

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

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 $(s, \ell) = (2, 3)$. Note that by Proposition 4.1, then

$$\tau_{\mathrm{Pow}}(2,3) = \frac{5}{8}n - \frac{3}{4}(k-1) - \frac{3}{8} \ ,$$

so these parameters allow the decoder to improve upon both half-the-minimum distance and the original Power decoding whenever the rate is between 1/6 and 1/2, for long enough codes.

Proposition 5.3. *Let $q = |\mathbb{F}|$. Whenever $d/2 \leq \epsilon < \tau_{\mathrm{Pow}}(2,3)$, the probability that Power decoding fails is upper bounded by*

$$\begin{cases} 4\big(q^{-8}\big)^{(\tau_{\mathrm{Pow}}(2,3)-\epsilon)-(0.29\epsilon/\log q - 1/4)} & \text{when } \epsilon \geq \frac{3}{5}n - \frac{4}{5}(k-1) \\ 4q^{-d/5+1+1.61\epsilon/\log q} & \text{when } \epsilon < \frac{3}{5}n - \frac{4}{5}(k-1) \end{cases} .$$

Proof. By Proposition 5.1, we can consider the probability over the choice of error vector, and simply bound the failure probability when the sent codeword was $\mathbf{0}$. Since we know by Proposition 5.2 that the failure probability is zero when $\epsilon < d/2$, then we can also assume $\epsilon \geq d/2$.

Fix now the number of errors ϵ and error positions $\mathcal{E}$, implying a specific Λ. For a given error $\boldsymbol{e} = \boldsymbol{r}$ with these non-zero positions, we will call $\boldsymbol{r}$, or R, "bad" if for R there exist λ_i, ψ_t solving Problem 3.2 and such that $\lambda_1 \neq \Lambda^s$ while $\deg \lambda_1 \leq \deg \Lambda^s$. Consequently, Power decoding fails only for bad error-values. Denote by $S_\Lambda \subset \mathbb{F}[x]$ the set of bad R. We will give an upper bound N on the size of S_Λ 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 Λ, and thus $N/(q-1)^\epsilon$ is a bound on the probability that Power decoding fails for any error of weight ϵ.

By assumption, the following equations are satisfied:

$$\begin{aligned} \psi_1 &= \lambda_1 R \ + \ \lambda_2 G \ , \\ \psi_2 &\equiv \lambda_1 R^2 + 2\lambda_2 RG \quad \bmod G^2 \ , \\ \psi_3 &\equiv \lambda_1 R^3 + 3\lambda_2 R^2 G \quad \bmod G^2 \ . \end{aligned}$$

Since $R(\alpha_i) = 0$ whenever $i \notin \mathcal{E}$, then $\Upsilon \mid R$ where $\Upsilon = G/\Lambda$. Thus the above implies $\Upsilon \mid \psi_1$ and $\Upsilon^2 \mid \psi_t$ for $t = 2,3$. Furthermore, letting $g \triangleq \gcd(\lambda_1, \Lambda)$, we can conclude that $g = \gcd(\psi_t, \Lambda)$ for all t. The regular form of the above three equations allows eliminating λ_1 and obtain:

$$\begin{aligned} \psi_2 - R\psi_1 &\equiv \lambda_2 RG \quad \bmod G^2 \ , \\ \psi_3 - R\psi_2 &\equiv \lambda_2 R^2 G \quad \bmod G^2 \ . \end{aligned}$$

From this we first note that $G \mid (\psi_2 - R\psi_1)$. We will use this fact momentarily. With the two above equations we continue to eliminate λ_2 and rewrite:

$$\begin{aligned} \psi_3 - R\psi_2 - R(\psi_2 - R\psi_1) &\equiv 0 \quad \bmod G^2 & \iff \\ R^2\psi_1 - 2R\psi_2 + \psi_3 &\equiv 0 \quad \bmod G^2 & \implies \\ R^2\psi_1^2 - 2R\psi_1\psi_2 + \psi_1\psi_3 &\equiv 0 \quad \bmod G^2 & \iff \\ (R\psi_1 - \psi_2)^2 + \psi_1\psi_3 &\equiv \psi_2^2 \quad \bmod G^2 \ . \end{aligned}$$

But we concluded just before that $G \mid (R\psi_1 - \psi_2)$ so $(R\psi_1 - \psi_2)^2 \equiv 0 \mod G^2$. This leaves the simple relation

$$\begin{aligned} \psi_2^2 &\equiv \psi_1\psi_3 \quad \bmod G^2 & \iff \\ \check{\psi}_2^2 \Upsilon &\equiv \check{\psi}_1\check{\psi}_3 \quad \bmod \Lambda^2 \ , \end{aligned} \tag{7}$$

where $\check{\psi}_t \triangleq \psi_t/\Upsilon^{\min(2,t)}$, and is a polynomial by our earlier observations. Thus, whenever R is bad, there is a triple $(\check{\psi}_1, \check{\psi}_2, \check{\psi}_3) \in \mathbb{F}[x]$ satisfying the above relation as well as

$$\deg \check{\psi}_t \leq d_t \triangleq 2\epsilon + t(k-1) - \min(2,t)(n-\epsilon) \ . \tag{8}$$

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 R could have the same triple. Recall that determining R up to congruence modulo Λ suffices, since this determines the error values. However, by our previous observation we have

$$\begin{aligned} R\psi_1 &\equiv \psi_2 \mod G && \iff \\ R\check{\psi}_1 &\equiv \check{\psi}_2 \Upsilon \mod \Lambda && \iff \\ R &\equiv (\check{\psi}_2\Upsilon/g)(\check{\psi}_1/g)^{-1} \mod \Lambda/g \ . \end{aligned}$$

This means that for a given triple $(\check{\psi}_t)_t$, having $\gcd(\check{\psi}_t, \Lambda) = g$, there can be at most $q^{\deg g}$ possible choices of R.

To bound the number of bad error values N for this given Λ, we will therefore perform a weighted count of all triples satisfying (7) and (8), where a triple is counted with weight $q^{\deg g}$, where g is a divisor of Λ dividing all the $\check{\psi}_t$:

$$\begin{aligned} N &\leq \sum_{g \mid \Lambda} q^{\deg g} \left| \left\{ (\check{\psi}_t)_t \in \mathbb{F}[x]^3 \ \middle| \ g \mid \check{\psi}_t,\ \deg \check{\psi}_t \leq d_t,\ \Upsilon\check{\psi}_2^2 \equiv \check{\psi}_1\check{\psi}_3 \mod \Lambda^2 \right\} \right| \\ &= \sum_{g \mid \Lambda} q^{\deg g} \left| \left\{ (\check{\check{\psi}}_t)_t \in \mathbb{F}[x]^3 \ \middle| \ \deg \check{\check{\psi}}_t \leq d_t - \deg g,\ \Upsilon\check{\check{\psi}}_2^2 \equiv \check{\check{\psi}}_1\check{\check{\psi}}_3 \mod (\Lambda/g)^2 \right\} \right| \ . \end{aligned}$$

Let T_g be the set inside the last sum. We use Lemma 5.4 (see below) to upper bound $|T_g|$, for any choice of g: setting $A = (\Lambda/g)^2$, $B = \Upsilon$ and $K_t = d_t - \deg g$ in that lemma, we get

$$|T_g| \leq 2^{\gamma + 2\epsilon - 2\deg g} q^{4\epsilon - (2n - 2(k-1)) + 1 + \max(0,\gamma) - \deg g} \ ,$$

where $\gamma = 5\epsilon - (3n - 4(k-1))$. This is only dependent on the *degree* of g. For each choice of $\deg g$, we can select g in $\binom{\epsilon}{\deg g}$ ways since $g \mid \Lambda$ and Λ splits into ϵ linear factors. This gives

$$N \leq q^{4\epsilon + 2n + 2(k-1) + 1 + \max(0,\gamma)} 2^{\gamma} \sum_{t=0}^{\epsilon} \binom{\epsilon}{t} 4^{\epsilon - t} \tag{9}$$

This can be simplified at a small loss of tightness. Firstly $\sum_{t=0}^{\epsilon} \binom{\epsilon}{t} 4^{\epsilon-t} = (4+1)^{\epsilon}$. For the case $\gamma \geq 0$, we rewrite into

$$N \leq q^{\epsilon + 8(\epsilon - \tau_{\mathrm{Pow}}(2,3)) - 2} 2^{\gamma} 5^{\epsilon} \qquad (\gamma \geq 0),$$

since $8\epsilon - (5n - 6(k-1) - 3) = 8(\epsilon - \tau_{\mathrm{Pow}}(2,3))$. Now $\gamma \leq \epsilon$, as can be seen as follows: since $\epsilon < \tau_{\mathrm{Pow}}(2,3)$ then $4\epsilon < \frac{5}{2}n - 3(k-1)$. Inserting in the expression for γ, we get that $\gamma \leq \epsilon - (n/2 - (k-1))$. But since we assumed $d/2 \leq \tau_{\mathrm{Pow}}(2,3)$ then $k - 1 \leq n/2$ which means $\gamma \leq \epsilon$. Therefore:

$$N \leq q^{\epsilon + 8(\epsilon - \tau_{\mathrm{Pow}}(2,3)) - 2} 10^{\epsilon} \qquad (\gamma \geq 0) \ .$$

For the case $\gamma < 0$, we instead get

$$N \leq q^{4\epsilon - 2d + 1} 5^{\epsilon} \qquad (\gamma < 0) \ .$$

Since $\gamma < 0$ then $\epsilon < \frac{3}{5}n - \frac{4}{5}(k-1) < \frac{3}{5}d$, and so $3\epsilon - 2d < -d/5$, which gives

$$N \leq q^{\epsilon - d/5 + 1} 5^{\epsilon} \qquad (\gamma < 0) \ .$$

As previously described $N/(q-1)^\epsilon$ then becomes a bound on the probability of decoding failure. The term $(\frac{q}{q-1})^\epsilon$ then appears, but $(\frac{q}{q-1})^\epsilon \leq (\frac{q}{q-1})^q \leq 2^2$. Finally, $10^\epsilon = (q^{-8})^{-\epsilon \log 10/(8 \log q)}$ and $\log 10/8 < 0.29$ and similarly for 5^ϵ. □

Lemma 5.4. *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 $Bf_2^2 \equiv f_1 f_3 \mod A$, while $\deg f_t \leq K_t$ and f_2 is monic. Then*

$$|S| \leq 2^{K_1+K_3} q^{K_2+1+\max(0,\gamma)} \ ,$$

where $\gamma = \max(K_1 + K_3,\ 2K_2 + \deg B) - \deg A$.

Proof. Consider first $\gamma < 0$ in which case $Bf_2^2 = f_1 f_3$. We can choose f_2 in q^{K_2-1} ways. The prime divisors of Bf_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 $\{Bf_2^2 + pA \mid p \in \mathbb{F}[x], \deg p \leq \gamma\}$, 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 5.3 demonstrates a rapid, exponential decrease in the probability of failure as the number of errors decrease away from $\tau_{\text{Pow}}(2,3)$. The bound only becomes non-trivial a few errors below $\tau_{\text{Pow}}(2,3)$, due to the term $0.29\epsilon/\log q - 1/4$ in the exponent. For instance, for a $[64, 27]$ code over $GF(64)$, then $\tau_{\text{Pow}}(2,3) = 20\,{}^1\!/\!{}_4$, but the bound is only less than 1 for $\epsilon \leq 19$, also for the unsimplified bound (9). Such a penalty is not observed in simulations, however, and seems to be an artefact of our proof. For the $[64, 27]$ code, decoding succeeds almost always with 20 errors (see next section). Similarly, for a $[256, 63]$ code over $GF(256)$, the bound is only non-trivial for $\epsilon < 108$, while (9) would be slightly better with $\epsilon < 110$; however, in simulations decoding works almost always up to $\lfloor \tau_{\text{Pow}}(2,3) \rfloor = 112$.

Nevertheless, in an asymptotic and relative sense, Proposition 5.3 guarantees that decoding up to $\tau_{\text{Pow}}(2,3)$ almost always succeeds:

Corollary 5.5. *When $s = 2$ and $\ell = 3$, with $n \to \infty$ while keeping q/n, k/n and ϵ/n constant, the probability that Power decoding fails goes to 0 when $\epsilon/n < \tau_{\text{Pow}}(2,3)/n$.*

Proof. Consider the high-error failure probability of Proposition 5.3:

$$4(q^{-8})^{(\tau_{\text{Pow}}(2,3)-\epsilon)-(0.29\epsilon/\log q - 1/4)} \leq 4(q^{-8n})^{\delta - 0.29\epsilon/(n \log q) + 1/(4n)} \ ,$$

where $\delta = \tau_{\text{Pow}}(2,3)/n - \epsilon/n$. Asymptotically δ approaches some positive constant, while the other terms in the exponent vanishes. The low-error case is similar. □

5.1. **Failure Behaviour in Relation to List Decoding.** It is natural to ask if the failure behaviour of Power decoding is linked to whether or not there are multiple codewords close to the received word, i.e. the list of codewords that e.g. the Guruswami–Sudan algorithm would return. There seems, however, to be no clear relation like this, as we explain below.

Consider that $\boldsymbol{c} \in \mathcal{C}$ was sent and $\boldsymbol{r}$ was received. Suppose Power decoding has a decoding radius of τ, and that we have a list decoder of the same decoding radius. Consider that $\boldsymbol{c}' \in \mathcal{C}$ is another codeword and assume that all other codewords are farther from $\boldsymbol{r}$ than $\boldsymbol{c}$ or $\boldsymbol{c}'$. Then there are the following possibilities:

1. $\mathrm{dist}(\boldsymbol{c}, \boldsymbol{r}) = \mathrm{dist}(\boldsymbol{c}', \boldsymbol{r}) \leq \tau$
2. $\mathrm{dist}(\boldsymbol{c}', \boldsymbol{r}) < \mathrm{dist}(\boldsymbol{c}, \boldsymbol{r}) \leq \tau$
3. $\mathrm{dist}(\boldsymbol{c}, \boldsymbol{r}) < \mathrm{dist}(\boldsymbol{c}', \boldsymbol{r}) \leq \tau$
4. $\mathrm{dist}(\boldsymbol{c}, \boldsymbol{r}) \leq \tau < \mathrm{dist}(\boldsymbol{c}', \boldsymbol{r})$
5. $\mathrm{dist}(\boldsymbol{c}', \boldsymbol{r}) \leq \tau < \mathrm{dist}(\boldsymbol{c}, \boldsymbol{r})$
6. $\tau < \mathrm{dist}(\boldsymbol{c}, \boldsymbol{r}), \mathrm{dist}(\boldsymbol{c}', \boldsymbol{r})$

Clearly, both Power decoding and the list decoder will fail in recovering $\boldsymbol{c}$ in Item 5 and Item 6. In Items 1–4, the list decoder guarantees to recover $\boldsymbol{c}$ on a list, though for Items 1–3 that list will have length at least 2.

For Power decoding it is less clear-cut. Firstly, for Items 1 and 2, then Power decoding "fails" according to the definition given at the beginning of Section 5. Indeed, for Item 2, then Power decoding guarantees to return $\boldsymbol{c}'$ or fail. For Item 1, however, then Algorithm 1 might be lucky and find $\boldsymbol{c}$, but in all likelihood the obtained solution to Problem 3.2 will be some linear combination of the two solutions corresponding to $\boldsymbol{c}$ and $\boldsymbol{c}'$; probably Line 4 or at least Line 5 of Algorithm 1 will return fail. For Items 3 and 4, then Power decoding will probably obtain $\boldsymbol{c}$; but in either case, one can construct examples where it will fail. That is, whether or not there is only one codeword within radius τ, then Power decoding might succeed or it might fail.

That might be surprising at first, so we give examples of these cases. Consider $\mathcal{C}$ to be the $[23, 7]_{GF(23)}$ GRS code with evaluation points $\boldsymbol{\alpha} = (0, 1, \ldots, 22)$ and $\boldsymbol{\beta} = \mathbf{1}$. For this code, $\tau_{\mathrm{Pow}}(2, 3) = 9$. As an example for Item 3 where Power decoding succeeds, consider the following vectors:

$$\boldsymbol{c}_3 = (16\ 15\ 20\ 20\ 3\ 0\ 18\ 0\ 19\ 16\ 2\ 11\ 11\ 3\ 9\ 18\ 5\ 0\ 0\ 0\ 5\ 0\ 16) \in \mathcal{C} \ ,$$
$$\boldsymbol{r}_3 = (16\ \ 0\ \ 20\ 20\ 0\ 0\ 18\ 0\ 19\ \ 0\ \ 2\ 11\ \ 0\ \ 0\ 0\ \ 0\ \ 5\ 0\ 0\ 0\ 5\ 0\ \ 0\ ) \ .$$

Then $8 = \mathrm{dist}(\boldsymbol{r}_3, \boldsymbol{c}_3) < \mathrm{dist}(\boldsymbol{r}_3, \mathbf{0}) = \tau = 9$. So a list decoder with decoding radius 9 would obtain the list $[\mathbf{0}, \boldsymbol{c}_3]$. Power decoding returns $\boldsymbol{c}_3$. Indeed, our observation is that Power decoding usually succeeds in the case of Item 3.

An example for Item 4 where Power decoding fails, even though there is only one nearby codeword, $\mathbf{0}$, is the following received word:

$$\boldsymbol{r}_4 = (0\ 2\ 9\ 1\ 0\ 0\ 6\ 0\ 0\ 0\ 5\ 0\ 0\ 0\ 0\ 0\ 0\ 4\ 8\ 15\ 0\ 0\ 12) \ .$$

This last example was found by random generation of error vectors of weight 9, after roughly 47 000 successful decoding trials. As an aside, the failure probability bound of Proposition 5.3 gives the trivial bound 1 for the failure probability in this case.

6. **Simulation Results.**

The proposed decoding algorithm has been conceptually implemented in Sage v8.0 [45], and is available for download at `http://jsrn.dk/code-for-articles`. The implementation follows the approach of Section 7, computing a solution basis using the Mulders–Storjohann algorithm [26]. The asymptotic complexity of the implementation is therefore $O(\ell^3 s^2 n^2)$.

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 τ, we have created $N = 10^5$ random errors of weight exactly τ and attempted to decode a received word $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ for some randomly chosen $\boldsymbol{c}$ (though, of course, Proposition 5.1 implies that shifting by $\boldsymbol{c}$ makes no difference). We have limited the decoding radii used to being $\lfloor \tau_{\mathrm{Pow}}(s, \ell) \rfloor + \{-1, 0, 1\}$. The results are listed as Table 1.

$[n,k]_q$	(s,ℓ)	τ_{Pow}	$P_f(\lfloor\tau_{\mathrm{Pow}}\rfloor - 1)$	$P_f(\lfloor\tau_{\mathrm{Pow}}\rfloor)$	$P_f(\lfloor\tau_{\mathrm{Pow}}\rfloor + 1)$	τ_{bnd}
$[21,3]_{23}$	$(6,19)$	$14\,1/120$	7.43×10^{-3}	1.97×10^{-1}	1	
$[24,7]_{25}$	$(2,3)$	$10\,1/4$	0	2.27×10^{-3}	1	8 (9)
$[32,10]_{37}$	$(2,4)$	13	0	2.78×10^{-2}	1	
$[64,27]_{64}$	$(2,3)$	$20\,1/4$	0	3.10×10^{-4}	1	19 (19)
$[68,31]_{71}$	$(3,4)$	20	0	0	1	
$[125,51]_{125}$	$(4,6)$	42	0	0	1	
$[256,63]_{256}$	$(2,4)$	$116\,2/5$	0	0	$1 - 3.00 \times 10^{-4}$	

TABLE 1. Simulation results. $P_f(\tau)$ denotes the observed probability of decoding failure (no result or wrong result) with random errors of weight exactly τ. τ_{bnd} indicates the number of errors ϵ for which Proposition 5.3 yields a bound < 1 (where applicable); in parentheses is if the probability estimate of (9) is used instead.

As is evident, $\tau_{\mathrm{Pow}}(s,\ell)$ very clearly describes the number of errors we can rely on correcting: the probability of failing appears to decay exponentially with $\tau_{\mathrm{Pow}}(s,\ell) - \epsilon$, as we might expect if extrapolating from the bound of Proposition 5.3. 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 $[21,3]_{GF(23)}$. For such a low-rate code, $\tau_{\mathrm{Pow}}(s,\ell)$ is quite close to the covering radius, and there is a significant probability that a random error will yield a received word which is closer to another codeword. 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 gave a 16.1% chance that another codeword was as close or closer to the sent codeword. Thus most of the 19.7% failures of Power decoding stem from this.

7. Efficient Solving of the Key Equations.

To solve Problem 3.2, we will leverage existing algorithms by modelling Problem 3.2 as a *simultaneous Hermite Padé approximation* (SH Padé), a well-studied computational problem. This problem does not fit perfectly to Problem 3.2, so to describe the modelling from one to the other we will introduce some technical notions pertaining to the solution sets of SH Padé problems. The upshot is Algorithm 2 and Corollary 7.10 stating that we can rely completely on existing sophisticated algorithms to solve Problem 3.2 in complexity $O^\sim(\ell^\omega n)$ (or the faster $O^\sim(s^2\ell^{\omega-1}n)$ if we rely on the unpublished [37]).

Definition 7.1 (Simultaneous Hermite Padé approximation)**.** Given $A \in \mathbb{F}[x]^{s\times\ell}$ and $\Gamma_1,\ldots,\Gamma_\ell \in \mathbb{F}[x]$, as well as degree bounds $T_i, N_t \in \mathbb{Z}_{\geq 0}$, compute, if it exists, $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_s) \in \mathbb{F}[x]^s$ such that $\deg \lambda_i < T_i$, and

$$\boldsymbol{\lambda} A \equiv \boldsymbol{\psi} \mod (\Gamma_1,\ldots,\Gamma_\ell) \ ,$$

where $\boldsymbol{\psi} = (\psi_1,\ldots,\psi_\ell) \in \mathbb{F}[x]^\ell$ satisfies $\deg \psi_t < N_t$. (The modulo operation is element-wise, i.e. the i'th entry of $\boldsymbol{\lambda} A$ is congruent to ψ_i modulo Γ_i.)

SH Padé approximations have appeared elsewhere in coding theory: for the interpolation step of Guruswami–Sudan and the Wu decoding algorithms for Reed–Solomon and other codes [10,53], and for decoding of Hermitian algebraic-geometry codes [29]. Computing solutions to these very general forms of Padé approximation goes back much further in the computer algebra community, though Γ_i are usually powers of x, see e.g. [4, 5] and the references therein. In the generality above, the problem was first considered in [31] solved using row reduction of $\mathbb{F}[x]$ matrices, and shortly thereafter in [10] solved as an $\mathbb{F}$-linear system exhibiting block-Hankel structure [9]. Even more general notions include minimal approximant basis, or order basis [15, 16], and relation bases [27].

First we define a measure of how far a solution is from the degree bounds:

Definition 7.2. For a given SH Padé problem with $A \in \mathbb{F}[x]^{s\times\ell}$ as well as Γ_i, T_i, N_t, and a vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_s)$, the *discrepancy* $\delta \in \mathbb{Z}$ of $\boldsymbol{\lambda}$ (wrt. the SH Padé problem) is given as:

$$\delta = \max\Big(\max_{i=1,\dots,s} (\deg \lambda_i - T_i),\ \max_{t=1,\dots,\ell} (\deg \psi_t - N_t) \Big) ,$$

where $\boldsymbol{\psi} = (\psi_1, \dots, \psi_\ell) = \boldsymbol{\lambda} A \ \operatorname{rem} (\Gamma_1, \dots, \Gamma_\ell)$.

Note that a $\boldsymbol{\lambda}$ is a solution to an SH Padé problem if and only if its discrepancy is negative. We wish to link the type of degree restrictions of Definition 7.1 with those of Problem 3.2: for this, observe that a solution $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_s) \in \mathbb{F}[x]^s$ has a discrepancy of $\deg \lambda_1 - T_1$ if and only if

$$\begin{aligned} \deg \lambda_1 - T_1 &\geq \deg \lambda_{i+1} - T_{i+1} && \text{for } i = 1, \dots, s-1 \\ \deg \lambda_1 - T_1 &\geq \deg \psi_t - N_t && \text{for } t = 1, \dots, \ell\ , \end{aligned}$$

where $\boldsymbol{\psi} = (\psi_1, \dots, \psi_\ell) = \boldsymbol{\lambda} A \ \operatorname{rem} (\Gamma_1, \dots, \Gamma_\ell)$. This leads to the following lemma:

Lemma 7.3. *Consider a received word $\boldsymbol{r}$, let $\tau \in \mathbb{Z}_{\geq 0}$ be a chosen decoding radius and assume at most τ errors occurred. Consider the SH Padé approximation defined by $A = [A_{i,t}] \in \mathbb{F}[x]^{s\times\ell}$ as well as Γ_t, T_i and N_t, given as:*

$$\begin{aligned} A_{i+1,t} &= \begin{cases} \binom{t}{i} R^{t-i} G^i & \text{for } t = 1, \dots, s-1 \text{ and } i = 0, \dots, s-1 \\ \binom{t}{i} R^{t-i} G^i \bmod G^s & \text{for } t = s, \dots, \ell \text{ and } i = 0, \dots, s-1 \end{cases} \\ \Gamma_t &= \begin{cases} x^{s\tau + t(n-1)+1} & \text{for } t = 1, \dots, s-1 \\ G^s & \text{for } t = s, \dots, \ell \end{cases} \\ T_{i+1} &= s\tau - i + 1 \quad \text{for } i = 0, \dots, s-1 \\ N_t &= s\tau + t(k-1) + 1 \quad \text{for } t = 1, \dots, \ell\ . \end{aligned}$$

Let $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_s)$ be a solution to this SH Padé approximation with minimal discrepancy among solutions whose discrepancy equals $\deg \lambda_1 - T_1$. Then $\boldsymbol{\lambda}, \boldsymbol{\psi}$ is a minimal solution to the instance of Problem 3.2 corresponding to $\boldsymbol{r}$, where $\boldsymbol{\psi} = \boldsymbol{\lambda} A \ \operatorname{rem} (\Gamma_1, \dots, \Gamma_\ell)$.

Proof. We first prove that $\boldsymbol{\lambda}$ is well-defined, that is there is such a solution to the SH Padé problem. We claim that if $\boldsymbol{\lambda}', \boldsymbol{\psi}'$ is a minimal solution to the instance of Problem 3.2 (which we know exists) then $\boldsymbol{\lambda}'$ is such a solution to the SH Padé approximation. Note first by the assumption on decoding radius that $\deg \lambda'_1 \leq s\tau$. Then Item 2 of Problem 3.2 implies $\deg \lambda'_{i+1} \leq s\tau - i < T_{i+1}$ for $i = 1, \dots, s-1$. Similarly, by Item 3 then $\deg \psi'_t \leq s\tau + t(k-1) < N_t$ for $t = 1, \dots, \ell$. Finally, observe that

$$\begin{aligned} \deg \lambda'_{i+1} + i \leq \deg \lambda'_1 &\iff \deg \lambda'_{i+1} - T_{i+1} \leq \deg \lambda'_1 - T_1 \\ \deg \psi'_t - t(k-1) \leq \deg \lambda'_1 &\iff \deg \psi'_t - N_t \leq \deg \lambda'_1 - T_1\ . \end{aligned} \tag{10}$$

so Item 2 and Item 3 of Problem 3.2 imply the discrepancy condition.

The other direction is very similar: assume now $\boldsymbol{\lambda}$ is a solution to the SH Padé problem with discrepancy $\deg \lambda_1 - T_1$, and $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_\ell) = \boldsymbol{\lambda} A \operatorname{rem} (\Gamma_1, \ldots, \Gamma_\ell)$. Item 1b of Problem 3.2 is obviously satisfied; for Item 1a, we know $\boldsymbol{\lambda} A_{*,t} \equiv \psi_t \mod \Gamma_t$ for $t = 1, \ldots, \ell$, where $A_{*,t}$ denotes the t'th column of A. Note that for $t < s$ we have

$$\deg(\boldsymbol{\lambda} A_{*,t}) \leq \max_{i=0,\ldots,s-1} \big((T_{i+1} - 1) + (\deg R^{t-i} G^i)\big) \leq s\tau + t(n-1) \ .$$

For these values of t we have $\Gamma_t = x^{s\tau + t(n-1)+1}$ and so the congruence lifts to equality, i.e. Item 1a. Item 2 and Item 3 of Problem 3.2, follow directly from the discrepancy condition on $\boldsymbol{\lambda}$ together with (10).

For minimality, assume conversely that $\deg \lambda'_1 < \deg \lambda_1$. Then $\deg \lambda'_1 - T_1 < \deg \lambda_1 - T_1$. Since $\boldsymbol{\lambda}'$ is a solution to the SH Padé problem satisfying the discrepancy restriction, then this contradicts the minimality of $\boldsymbol{\lambda}$. □

7.1. **Solution bases for Padé approximations.**

Lemma 7.3 states that special solutions to a specific SH Padé problem are actually minimal solutions to Problem 3.2. Many algorithms for solving SH Padé problems, and in particular the fastest ones known, actually find a basis of *all* solutions, for a notion of "basis" which we introduce momentarily. We will now show that such a basis must contain a solution satisfying the constraints of Lemma 7.3 and hence will be a minimal solution to Problem 3.2. This section uses a number of concepts which are standard in polynomial matrix literature, but less so in coding theory. They will not be used outside this section.

The degree of a vector $\boldsymbol{v} \in \mathbb{F}[x]^m$ or matrix $A \in \mathbb{F}[x]^{m' \times m}$ is the maximal degree of its entries. The *leading matrix* of A, denoted $\mathrm{LM}(A) \in \mathbb{F}^{m' \times m}$, has (i,j)'the entry equal to the coefficient of x^{d_i} of $A_{i,j}$, where d_i is the degree of the i'th row of A. The *leading indices* of $\boldsymbol{v}$, denoted $\mathrm{leads}(\boldsymbol{v}) \subset \{1, \ldots, m\}$, are the indices of $\boldsymbol{v}$ which have degree $\deg \boldsymbol{v}$. In other words, $\mathrm{LM}(A)$ is non-zero exactly at the leading indices of the rows of A. We also introduce *shifted* variants of the above notions: given a "shift" $\boldsymbol{h} \in \mathbb{Z}^m$, then $\deg_{\boldsymbol{h}} \boldsymbol{v} := \deg(\boldsymbol{v} x^{\boldsymbol{h}})$, where $x^{\boldsymbol{h}}$ is the diagonal matrix with entries $x^{h_1}, \ldots, x^{h_m}$. Similarly $\deg_{\boldsymbol{h}} A := \deg(A x^{\boldsymbol{h}})$; $\mathrm{LM}_{\boldsymbol{h}}(A) := \mathrm{LM}(A x^{\boldsymbol{h}})$; and $\mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{v}) := \mathrm{leads}(\boldsymbol{v} x^{\boldsymbol{h}})$. Note that if $\boldsymbol{h}$ has negative entries, this notation may formally pass over the ring of Laurent series.

To relate some of these concepts to the previous section, note that if we set $\boldsymbol{h} = (-T_1, \ldots, -T_s, -N_1, \ldots, -N_\ell)$, then for any $\boldsymbol{\lambda} \in \mathbb{F}[x]^s$ the discrepancy of $\boldsymbol{\lambda}$ is exactly $\deg_{\boldsymbol{h}}(\boldsymbol{\lambda} | \boldsymbol{\psi})$, where $\boldsymbol{\psi} = \boldsymbol{\lambda} A \operatorname{rem} (\Gamma_1, \ldots, \Gamma_\ell)$. Further, the requirement of Lemma 7.3, i.e. that the discrepancy of $\boldsymbol{\lambda}$ equals $\deg \lambda_1 - T_1$, is equivalent to requiring $1 \in \mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{\lambda} | \boldsymbol{\psi})$.

Lemma 7.4. *Consider a given SH Padé problem with $A \in \mathbb{F}[x]^{s \times \ell}$ as well as Γ_t, T_i and N_t. A vector $\boldsymbol{\lambda} \in \mathbb{F}[x]^s$ is a solution iff there is $\boldsymbol{\psi} \in \mathbb{F}[x]^\ell$ such that $\deg_{\boldsymbol{h}}(\boldsymbol{\lambda} | \boldsymbol{\psi}) < 0$ and $(\boldsymbol{\lambda} | \boldsymbol{\psi})$ is in the row space of M, where*

$$M = \left[\begin{array}{c|c} I_{s \times s} & A \\ \hline & \mathrm{diag}(\Gamma_1, \ldots, \Gamma_\ell), \end{array}\right]$$
$$\boldsymbol{h} = (-T_1, \ldots, -T_s, -N_1, \ldots, -N_\ell) \ .$$

Proof. The row space of M is exactly the vectors $(\boldsymbol{\lambda} | \boldsymbol{\psi})$ where $\boldsymbol{\lambda} A \equiv \boldsymbol{\psi} \mod (\Gamma_1, \ldots, \Gamma_\ell)$. For such vectors, then $\boldsymbol{\lambda}$ is solution when it has negative discrepancy, which by our earlier observation is exactly when it has negative $\boldsymbol{h}$-degree. □

We say that matrix A is $\boldsymbol{h}$*-row reduced* if $\mathrm{LM}_{\boldsymbol{h}}(A)$ has full row rank, see [19, Ch. 6.3.2] and [4]. For any $M \in \mathbb{F}[x]^{m' \times m}$ with full row rank, there always exists another matrix A which is $\boldsymbol{h}$-row reduced and has the same row space as M; see e.g. [26] for a succinct iterative algorithm. Row reduced matrices derive their interest from having minimal row degrees of all possible bases of the same row space; this property can be generalised to the *predictable degree property* [19, Ch. 6.3.2], of which we will use the following variant:

Proposition 7.5. *Let $\boldsymbol{h} \in \mathbb{Z}^m$ be a shift, let $A \in \mathbb{F}[x]^{m' \times m}$ be row reduced, and let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m'}$ be the rows of A. Let $\boldsymbol{v} \in \mathbb{F}[x]^m$ be any vector in the row space of A. Then there exists $\boldsymbol{q} = (q_1, \ldots, q_{m'}) \in \mathbb{F}[x]^{m'}$ such that $\boldsymbol{v} = \boldsymbol{q}A$ and*

$$\deg_{\boldsymbol{h}} \boldsymbol{v} = t \ , \ \textit{where } t = \max_{i=1,\ldots,m'} (\deg q_i + \deg_{\boldsymbol{h}} \boldsymbol{a}_i), \quad \textit{and}$$

$$\mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{v}) \subseteq \bigcup_{i \in I} \mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{a}_i) \ , \ \textit{where } I = \{ i \mid \deg q_i = \deg_{\boldsymbol{h}} \boldsymbol{v} - \deg_{\boldsymbol{h}} \boldsymbol{a}_i \} \ .$$

Proof. The existence of $\boldsymbol{q}$ is trivial since $\boldsymbol{v}$ is in the row space of A. Note that

$$\boldsymbol{v} x^{\boldsymbol{h}} = \boldsymbol{q} A x^{\boldsymbol{h}} \ , \tag{11}$$

and so $\deg_{\boldsymbol{h}} \boldsymbol{v} \leq t$. Let $\grave{\boldsymbol{q}} \in \mathbb{F}^m$ be the scalar vector whose i'th entry is the leading coefficient of q_i if $i \in I$ and 0 otherwise. Let $\grave{\boldsymbol{v}} \in \mathbb{F}^m$ be the scalar vector of x^t'th coefficients of $\boldsymbol{v} x^{\boldsymbol{h}}$. Then (11) implies $\grave{\boldsymbol{v}} = \grave{\boldsymbol{q}} \mathrm{LM}_{\boldsymbol{h}}(A)$. But then $\grave{\boldsymbol{v}} \neq \boldsymbol{0}$ since $\mathrm{LM}_{\boldsymbol{h}}(A)$ has full row rank. Therefore $\deg_{\boldsymbol{h}}(\boldsymbol{v}) = t$. Further, $\mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{v})$ is then the indices of non-zero entries of $\grave{\boldsymbol{v}}$, i.e. the non-zero entries of $\grave{\boldsymbol{q}} \mathrm{LM}_{\boldsymbol{h}}(A)$, i.e. a subset of the non-zero columns of $\mathrm{LM}_{\boldsymbol{h}}(A')$, where A' is the rows of A indexed by I. □

We are now in a position to define a notion of "basis" of all solutions:

Definition 7.6. Consider a given SH Padé problem with $A \in \mathbb{F}[x]^{s \times \ell}$ as well as Γ_t, T_i and N_t. Let B' be any matrix which is left-equivalent[3] to M and $\boldsymbol{h}$-row reduced, where M and $\boldsymbol{h}$ are as in Lemma 7.4. Let $B \in \mathbb{F}[x]^{m \times (s+\ell)}$ consist of the rows of B' with negative $\boldsymbol{h}$-degree. Then B is a *solution basis* to the SH Padé problem.

Not only will the rows of a solution basis B be solutions themselves; the main point is that they will *span* every single solution in a predictable way: any solution must be a linear combination of the rows of the complete, $\boldsymbol{h}$-row reduced matrix B', but due to the predictable degree property, for the $\boldsymbol{h}$-degree of a vector to be negative, it must be spanned only by vectors with negative $\boldsymbol{h}$-degree themselves, and with coefficients of bounded degree.

We now see an easy algorithm for solving SH Padé problems: set up M and compute a row reduced matrix B' which is left-equivalent to M. This could e.g. be done using the iterative Mulders–Storjohann algorithm [46], or using the reduction from row reduction to order basis [15,16]. The latter yields a complexity of $O^{\sim}((s+\ell)^{\omega} D)$, where $D = \max T_i + \max \deg g_t$.

We will continue the discussion a bit further, since a result from [37] – which is not yet published – allows a faster algorithm if we only compute the first s columns of a solution basis:

Definition 7.7. Consider a given SH Padé problem with $A \in \mathbb{F}[x]^{s \times \ell}$ as well as Γ_t, T_i and N_t. A *solution specification* is a matrix $L \in \mathbb{F}[x]^{m \times s}$ and discrepancies

[3]I.e. there exists an invertible matrix $U \in \mathbb{F}[x]^{(s+\ell) \times (s+\ell)}$ such that $B' = UM$.

Algorithm 2 Solving Problem 3.2 using SH Padé approximation

```
Input: $R, G \in \mathbb{F}[x]$, $s, \ell, \tau \in \mathbb{Z}_+$ with $s \leq \ell$.
Output: A minimal solution $(\boldsymbol{\lambda}|\boldsymbol{\psi})$ to Problem 3.2 if one exist, or fail
1 Compute $A \in \mathbb{F}[x]^{s \times \ell}$ as in Lemma 7.3, and set $\Gamma_t, T_{i+1}, N_t$ as in that lemma.
2 $L, \boldsymbol{\delta} \leftarrow$ solution specification to the SH Padé problem of $A$ and $\Gamma_t, T_{i+1}, N_t$.
3 $\boldsymbol{\lambda} \leftarrow$ a minimal $\boldsymbol{h}$-degree row of $L$ among rows with $1 \in \mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{\lambda})$, where $\boldsymbol{h}$ is
  as in Lemma 7.4. If there is no such row, return fail.
4 $\boldsymbol{\psi} \leftarrow \boldsymbol{\lambda} A \ \mathrm{rem}\ (\Gamma_1, \dots, \Gamma_\ell)$.
5 return $(\boldsymbol{\lambda}|\boldsymbol{\psi})$.
```

$\delta_1, \dots, \delta_m < 0$ such that there is a matrix $\grave{B} \in \mathbb{F}[x]^{m \times \ell}$ for which $[L \mid \grave{B}]$ is a solution basis, and whose rows have $\boldsymbol{h}$-degree $\delta_1, \dots, \delta_m$.

Proposition 7.8 ([37])**.** *Consider an SH Padé approximation problem with $A \in \mathbb{F}[x]^{s \times \ell}$ as well as Γ_i, T_i, N_t, satisfying $s < \ell$, and $T_i < \deg \mathrm{lcm}(\Gamma_1, \dots, \Gamma_\ell)$ for $i = 1, \dots, s$, and $\deg N_t < \deg \Gamma_t$ for $t = 1, \dots, \ell$. There exists an algorithm which computes a solution specification using*

$$O(\ell^{\omega-1}\mathsf{M}(sD)(\log sD)(\log sD/\ell)^2) \subset O^{\sim}(s\ell^{\omega-1}D)$$

operations in $\mathbb{F}$, where $D = \max_i T_i + \max_t \deg \Gamma_t$.

To solve Problem 3.2 using SH Padé approximations in the complexity of Proposition 7.8, the only remaining piece is to prove that a solution specification must contain a row for which we can apply Lemma 7.3.

Proposition 7.9. *Consider an SH Padé approximation problem with $A \in \mathbb{F}[x]^{s \times \ell}$ as well as Γ_i, T_i, N_t. If there exists a solution $\boldsymbol{\lambda} \in \mathbb{F}[x]^s$ such that its discrepancy equals $\deg \lambda_1 - T_1$, then such a solution with minimal discrepancy will appear in a solution specification.*

Proof. Let $B \in \mathbb{F}[x]^{(s+\ell)\times(s+\ell)}$ be a completed, $\boldsymbol{h}$-row reduced matrix, left-equivalent to M, corresponding to the solution specification, where $\boldsymbol{h}$ and M are as in Lemma 7.4. Let $\boldsymbol{\lambda} \in \mathbb{F}[x]^s$ be a solution with minimal discrepancy satisfying $\deg \lambda_1 - T_1$. Then there is $\boldsymbol{\psi} \in \mathbb{F}[x]^\ell$ and $\boldsymbol{q} = (q_1, \dots, q_{s+\ell}) \in \mathbb{F}[x]^{s+\ell}$ such that $(\boldsymbol{\lambda}|\boldsymbol{\psi}) = \boldsymbol{q}B$ and $\deg_{\boldsymbol{h}}(\boldsymbol{\lambda}|\boldsymbol{\psi}) = \deg \lambda_1 - T_1$, i.e. $1 \in \mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{\lambda}|\boldsymbol{\psi})$. By Proposition 7.5 then there is a row $\boldsymbol{b}_i$ of B with $1 \in \mathrm{leads}_{\boldsymbol{h}}(\boldsymbol{\lambda}|\boldsymbol{\psi})$ and $q_i \neq 0$. Then $\deg_{\boldsymbol{h}} \boldsymbol{b}_i \leq \deg_{\boldsymbol{h}}(\boldsymbol{\lambda}|\boldsymbol{\psi})$, so if we write $\boldsymbol{b}_i = (\boldsymbol{\lambda}'|\boldsymbol{\psi}')$ then $\boldsymbol{\lambda}'$ is a solution with discrepancy $\deg \lambda'_1 - T_1 \leq \deg \lambda_1 - T_1$. To not contradict our choice of $\boldsymbol{\lambda}$, then equality must hold, and $\boldsymbol{\lambda}'$ is a satisfactory solution appearing in a solution specification. □

A complete algorithm for solving Problem 4 using solution specifications of SH Padé approximations is given as Algorithm 2.

Corollary 7.10. *Algorithm 2 is correct. It has complexity $O(\ell^\omega \mathsf{M}(sn)(\log sn)) \subset O^{\sim}(s\ell^\omega n)$ operations in $\mathbb{F}$, using [15, 16] for Line 2. Using the results of [38], it has complexity $O(\ell^{\omega-1}\mathsf{M}(s^2n)(\log sn)(\log sn/\ell)^2) \subset O^{\sim}(s^2\ell^{\omega-1}n)$.*

The same expressions hold for the complexity of Algorithm 1.

Proof. Correctness follows from the results of this section and the last; note that the requirements of Proposition 7.8 are satisfied in our case and that $D \in O(sn)$. For complexity, we merely need to argue that computing the solution specification of the SH Padé approximation dominates. Indeed, A can be computed using dynamic programming in $O(\ell s \mathsf{M}(sn))$ by remarking that $R^{t-i}G^i$ can be computed as the product of two previously computed terms of roughly half the size. The only other non-trivial computation is that of $\boldsymbol{\psi}$ which can also be carried out in complexity $O(\ell s \mathsf{M}(sn))$. $\square$

Remark 7.11. For short block-lengths, it can be of interest to consider the computational complexity when not using fast arithmetic, i.e. taking $\mathsf{M}(n) = O(n^2)$ and $n^\omega = O(n^3)$. In this regime, a much simpler algorithm than those mentioned in Corollary 7.10 is to compute a solution basis by applying the Mulders–Storjohann row-reduction algorithm [26]. This yields the complexity $O(\ell^3 s^2 n^2)$, which is similar to complexities of interpolation algorithms for the Guruswami–Sudan in this regime, see e.g. [22, 34, 53].

8. **Re-Encoding.**

"Re-Encoding" is a simple technique invented by Kötter and Vardy, originally for reducing the computational burden of the interpolation step in the Guruswami–Sudan algorithm [21]. It is especially powerful when using different multiplicities at each point, such as in the Kötter–Vardy soft-decision decoding version of Guruswami–Sudan [20]. 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 $\hat{\boldsymbol{r}}$ is the received word. Using Lagrange interpolation, we can easily compute the unique $\hat{\boldsymbol{c}} = \mathrm{ev}(\hat{f}) \in \mathcal{C}$ such that $\hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{r}}$ coincide on the first k positions. Clearly, decoding $\boldsymbol{r} = \hat{\boldsymbol{r}} - \hat{\boldsymbol{c}}$ immediately gives a decoding of $\hat{\boldsymbol{r}}$, and thanks to Proposition 5.1 we know Power decoding will succeed on $\boldsymbol{r}$ if and only if it succeeds on $\hat{\boldsymbol{r}}$. The idea of re-encoding is that the leading k zeroes of the resulting $\boldsymbol{r}$ might be utilised in the decoding procedure to reduce the computation cost of decoding $\boldsymbol{r}$.

Assume therefore for this section that $\boldsymbol{r}$ is the received word after re-encoding and therefore has k leading zeroes. That means $\hat{G} \mid R$ where $\hat{G} = \prod_{i=1}^{k}(x - \alpha_i)$. Consider the linearised key equations of Problem 3.2. Each of them are now divisible by $\hat{G}^{\min(s,t)}$, and so we obtain:

$$1a') \quad \psi_t/\hat{G}^t = \sum_{i=0}^{t} \lambda_{i+1} \cdot \left(\binom{t}{i} R^{t-i} G^i / \hat{G}^t \right) , \qquad \text{for } t = 1, \dots, s-1$$

$$1b') \quad \psi_t/\hat{G}^s \equiv \sum_{i=0}^{s-1} \lambda_{i+1} \cdot \left(\binom{t}{i} R^{t-i} G^i / \hat{G}^s \right) \mod (G/\hat{G})^s , \quad \text{for } t = s, \dots, \ell .$$

The elements $\grave{\psi}_t \triangleq \psi_t/\hat{G}^{\min(s,t)}$ and $R^{t-i}G^i/\hat{G}^{\min(s,t)}$ are all polynomials, but of much lower degree than before. Thus, we can solve for λ_i and $\grave{\psi}_t$ directly which has fewer coefficients. The degree restriction on $\grave{\psi}_t$ becomes

$$\deg \lambda_1 + t(k-1) - \min(s,t)k \geq \deg \grave{\psi}_t .$$

The complete decoding algorithm is exactly as Algorithm 1 with Line 3 replaced by the re-encoded key equations, and where f in Line 4 can be computed as $f = \grave{\psi}_1 \hat{G}/\lambda_1$.

To solve the re-encoded key equations, we proceed exactly as before: the following is an analogue of Lemma 7.3 linking the restrictions on λ_i and $\grave{\psi}_t$ to an SH Padé approximation, whose proof is analogous to that of Lemma 7.3.

Lemma 8.1. *Consider a received word $\boldsymbol{r}$ whose first k positions are 0, let $\tau \in \mathbb{Z}_{\geq 0}$ be a chosen decoding radius and assume at most τ errors occurred. Consider the SH Padé approximation defined by $A = [A_{i,t}] \in \mathbb{F}[x]^{s \times \ell}$ as well as Γ_t, T_i and N_t, given as:*

$$A_{i+1,t} = \begin{cases} \binom{t}{i} R^{t-i} G^i / \hat{G}^t & \text{for } t = 1, \dots, s-1 \text{ and } i = 0, \dots, s-1 \\ \binom{t}{i} R^{t-i} G^i / \hat{G}^s \bmod (G/\hat{G})^s & \text{for } t = s, \dots, \ell \text{ and } i = 0, \dots, s-1 \end{cases}$$

$$\Gamma_t = \begin{cases} x^{s\tau + t(n-k-1)+1} & \text{for } t = 1, \dots, s-1 \\ (G/\hat{G})^s & \text{for } t = s, \dots, \ell \end{cases}$$

$$T_{i+1} = s\tau - i + 1 \quad \text{for } i = 0, \dots, s-1$$

$$N_t = s\tau + t(k-1) - \min(s,t)k + 1 \quad \text{for } t = 1, \dots, \ell \ .$$

Let $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_s)$ be a solution to this SH Padé approximation with minimal discrepancy among solutions whose discrepancy equals $\deg \lambda_1 - T_1$. Then $\boldsymbol{\lambda}, \grave{\boldsymbol{\psi}}$ is a minimal solution to the instance of Problem 3.2 corresponding to $\boldsymbol{r}$, where $\grave{\boldsymbol{\psi}} = \boldsymbol{\lambda} A \ \operatorname{rem} (\Gamma_1, \dots, \Gamma_\ell)$.

As mentioned, the asymptotic complexity of solving the SH Padé approximation of Lemma 8.1 is not lower than that of Corollary 7.10, in the usual asymptotic regime where we take $k \in \Theta(n)$. However, considering Proposition 7.8, the expression D becomes $s(\tau + n - k)$ rather than $s(\tau + n)$. This should give a noticeable, constant factor speed-up for the complete decoding algorithm.

9. Syndrome Key Equations.

As described in Section 2, the first key equation decoding algorithm was based on the notion of syndrome polynomial [8], and similarly, Power decoding without multiplicities was first described using a similar list of key equations [42]. The key equations of Theorem 3.1 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 $\mathsf{rev}_d(p)$ which we defined in Section 2.2. 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:

$$R^{(i,t)} \triangleq R^{t-i} \bmod G^{s-i} \qquad\qquad S^{(i,t)} = \frac{\mathsf{rev}(R^{(i,t)})}{\mathsf{rev}(G)^{s-i}} \ .$$

Note the degree that the reversal-operator on $\mathsf{rev}(R^{(i,t)})$ 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 $S^{(1,1)}$ equals the classical syndrome polynomial S which we used in Section 2.2, and $S^{(1,t)}$ equals the syndromes $S^{(t)}$ discussed in Section 2.3. The syndromes $S^{(i,t)}$ also appear (with a slightly different definition) in the Interpolation key equations for Guruswami–Sudan by Gentner et al. [53]. We can then formulate the—markedly more involved—syndrome variant of Theorem 3.1:

Theorem 9.1. *For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then there exist $g_t \in \mathbb{F}[x]$ for $t = s, \dots, \ell$ such that*

$$\begin{aligned}
\sum_{i=0}^{t} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}S^{(i,t)}\right) &\equiv 0 \mod x^{\varrho_t} && \textit{for } t = 1, \dots, s-1 \ , \\
\sum_{i=0}^{t} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}S^{(i,t)}x^{\iota_{i,t}}\right) &\equiv g_t \mod x^{\varrho_t} && \textit{for } t = s, \dots, \ell \ ,
\end{aligned}$$

where

$$\begin{aligned}
\deg g_t &\leq \begin{cases} \epsilon s - s & \textit{if } t = s \\ \epsilon s - 1 & \textit{if } t > s \end{cases} \\
\varrho_t &= \begin{cases} t(n-k) & \textit{if } t \leq s \\ sn - t(k-1) - 1 & \textit{otherwise} \end{cases} \\
\iota_{i,t} &= \begin{cases} 0 & \textit{if } t = s \\ i & \textit{if } t > s \end{cases} .
\end{aligned}$$

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 3.1 gives us

$$\begin{aligned}
\textstyle\sum_{i=0}^{t} \left(\Lambda^{s-i}\Omega^i\right)\left(\binom{t}{i}R^{(i,t)}G^i\right) &= \Lambda^s f^t && \iff \\
\mathsf{rev}_{\epsilon s + t(n-1)}\left(\textstyle\sum_{i=0}^{t} \left(\Lambda^{s-i}\Omega^i\right)\left(\binom{t}{i}R^{(i,t)}G^i\right)\right) &= \mathsf{rev}_{\epsilon s + t(n-1)}(\Lambda^s f^t) \ ,
\end{aligned}$$

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

$$\begin{aligned}
\textstyle\sum_{i=0}^{t} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}\mathsf{rev}(R^{(i,t)})\mathsf{rev}(G)^i\right) &= \mathsf{rev}(\Lambda^s f^t)x^{t(n-k)} && \Longrightarrow \\
\textstyle\sum_{i=0}^{t} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}\mathsf{rev}(R^{(i,t)})\mathsf{rev}(G)^i\right) &\equiv 0 \mod x^{t(n-k)} && \iff \\
\textstyle\sum_{i=0}^{t} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}S^{(i,t)}\right) &\equiv 0 \mod x^{t(n-k)} \ ,
\end{aligned}$$

where the last line follows from $\mathsf{rev}(G)^s$ 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 3.1, we can readily replace $R^{t-i}G^i$ with $R^{(i,t)}G^i$ modulo G^s. This gives us:

$$\begin{aligned}
\Lambda^s f^t &\equiv \sum_{i=0}^{s-1} \left(\Lambda^{s-i}\Omega^i\right)\left(\binom{t}{i}R^{(i,t)}G^i\right) \mod G^s && \Longrightarrow \\
\Lambda^s f^t + \mathsf{rev}(g_t)G^s &= \sum_{i=0}^{s-1} \left(\Lambda^{s-i}\Omega^i\right)\left(\binom{t}{i}R^{(i,t)}G^i\right) \ ,
\end{aligned}$$

for some $\mathsf{rev}(g_t) \in \mathbb{F}[x]$. The degree of the right-hand side is bounded as

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

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:

$$\begin{aligned}
\mathsf{rev}(\Lambda^s f^t)x^{\varrho_t} + g_t\mathsf{rev}(G)^s &= \sum_{i=0}^{s-1} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}\mathsf{rev}(R^{(i,t)})\mathsf{rev}(G)^i x^{\iota_{i,t}}\right) && \Longrightarrow \\
g_t\mathsf{rev}(G)^s &\equiv \sum_{i=0}^{s-1} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}\mathsf{rev}(R^{(i,t)})\mathsf{rev}(G)^i x^{\iota_{i,t}}\right) \mod x^{\varrho_t} && \Longleftrightarrow \\
g_t &\equiv \sum_{i=0}^{s-1} \mathsf{rev}(\Lambda^{s-i}\Omega^i)\left(\binom{t}{i}S^{(i,t)} x^{\iota_{i,t}}\right) \mod x^{\varrho_t} \ .
\end{aligned}$$

□

Remark 9.2. Just as we remarked that the key equations of Theorem 3.1 resemble certain characterisations of Interpolation polynomials in Guruswami–Sudan, so does the above syndrome formulation resemble the syndrome Interpolation key equations of [53]. Again, the deeper relation between the error locator approach and the Guruswami–Sudan is unclear.

Theorem 9.1 leads to a decoding algorithm in the very same way as Theorem 3.1. We could call these algorithms “Power syndromes” and “Power Gao” respectively. We have the following important remark:

Corollary 9.3. *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 9.1: 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.

We won’t go through the details of a syndrome-decoding analogue of Lemma 7.3, but it follows along exactly the lines we have seen in both Section 7 and Section 8. The asymptotic complexity is again the same as that of Corollary 7.10, but one would again expect a noticeable, constant factor speed-up very similar to that of the re-encoding transformation.

10. **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 [18]. The resulting decoder is, as the original Power decoding algorithm of Schmidt, Sidorenko and Bossert [42], a partial decoder which fails for a few error patterns within its decoding radius. We showed how one can efficiently solve the resulting key equations using existing algorithms for simultaneous Hermite Padé approximation problems.

The proposed decoder has applications as a simpler alternative to the Guruswami–Sudan algorithm—especially for hardware implementations—due to its simple one-step shift-register type structure. In particular, for medium rate codes, Power decoding with a low multiplicity of 2 or 3 would not be much more complicated to implement than half-the-minimum distance decoding while still offering a significant improvement in decoding performance. For such parameters, the root-finding step of the Guruswami–Sudan algorithm would have a significant circuit area and latency.

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 5.1 simplifies the equations one needs to analyse, and this was instrumental in the case for which we were able to bound the failure probability: $(s, \ell) = (2, 3)$. For these parameters, the decoding radius improves upon the case $s = 1$ whenever the rate is within $]1/6; 1/2[$. The claimed decoding radius of the decoder for other parameters was backed by simulations on a range of codes: this demonstrates a failure probability which seems to decay exponentially 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.

The proposed decoding algorithm has already been adapted to improve decoding of Interleaved RS codes [36]. Power decoding has previously been applied to other codes as well, e.g. Complex RS codes [25], and it seems clear that the proposed addition of multiplicities can aid those applications as well. Another interesting question is to extend Power decoding to soft-decision decoding, similar to Kötter–Vardy's variant of the Guruswami–Sudan algorithm [20].

11. **Acknowledgements.** The author would like to thank Vladimir Sidorenko, Martin Bossert and Daniel Augot for discussions on Power decoding and this paper. The author gratefully acknowledges the support of the Digiteo foundation, project IdealCodes while he was with Inria, 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

[1] A. Ahmed, R. Koetter, and N. R. Shanbhag. VLSI Architectures for Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Trans. Inf. Theory*, 57(2):648–667, Feb. 2011.

[2] M. Alekhnovich. Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed–Solomon Codes. *IEEE Trans. Inf. Theory*, 51(7):2257–2265, July 2005.

[3] G. Baker and P. Graves-Morris. *Padé approximants*, volume 59. Cambridge Univ. Press, 1996.

[4] M. V. Barel and A. Bultheel. A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numerical Algorithms*, 3(1):451–461, Dec. 1992.

[5] B. Beckermann and G. Labahn. A Uniform Approach for the Fast Computation of Matrix-Type Padé Approximants. *SIAM J. Matr. Anal. Appl.*, 15(3):804–823, July 1994.

[6] P. Beelen and K. Brander. Key equations for list decoding of Reed–Solomon codes and how to solve them. *J. Symb. Comp.*, 45(7):773–786, 2010.

[7] P. Beelen, T. Høholdt, J. S. R. Nielsen, and Y. Wu. On Rational Interpolation-Based List-Decoding and List-Decoding Binary Goppa Codes. *IEEE Trans. Inf. Theory*, 59(6):3269–3281, June 2013.

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

[9] A. Bostan, C.-P. Jeannerod, and E. Schost. Solving structured linear systems with large displacement rank. *Th. Comp. Sc.*, 407(1–3):155–181, Nov. 2008.

[10] M. Chowdhury, C.-P. Jeannerod, V. Neiger, E. Schost, and G. Villard. Faster Algorithms for Multivariate Interpolation With Multiplicities and Simultaneous Polynomial Approximations. *IEEE Trans. Inf. Theory*, 61(5):2370–2387, May 2015.

[11] H. Cohn and N. Heninger. Ideal forms of Coppersmith's theorem and Guruswami–Sudan list decoding. *arXiv*, 1008.1284, 2010.

[12] H. Cohn and N. Heninger. Approximate common divisors via lattices. *Proc. of ANTS*, 1(1):271–293, 2012.

[13] P. Fitzpatrick. On the Key Equation. *IEEE Trans. Inf. Theory*, 41(5):1290–1302, 1995.

[14] S. Gao. A New Algorithm for Decoding Reed-Solomon Codes. In *Communications, Information and Network Security*, number 712 in S. Eng. and Comp. Sc., pages 55–68. Springer, Jan. 2003.

[15] P. Giorgi, C. Jeannerod, and G. Villard. On the Complexity of Polynomial Matrix Computations. In *Proc. of ISSAC*, pages 135–142, 2003.

[16] S. Gupta, S. Sarkar, A. Storjohann, and J. Valeriote. Triangular -basis decompositions and derandomization of linear algebra algorithms over. *J. Symb. Comp.*, 47(4):422–453, Apr. 2012.

[17] V. Guruswami and M. Sudan. Improved Decoding of Reed–Solomon Codes and Algebraic-Geometric Codes. *IEEE Trans. Inf. Theory*, 45(6):1757–1767, 1999.

[18] S. M. Johnson. A New Upper Bound for Error-Correcting Codes. *IEEE Trans. Inf. Theory*, 46:203–207, 1962.

[19] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[20] R. Kötter and A. Vardy. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Trans. Inf. Theory*, 49(11):2809–2825, 2003.

[21] R. Kötter and A. Vardy. A Complexity Reducing Transformation in Algebraic List Decoding of Reed–Solomon Codes. In *Proc. of IEEE ITW*, 2003.

[22] K. Lee and M. E. O'Sullivan. List Decoding of Reed–Solomon Codes from a Gröbner Basis Perspective. *J. Symb. Comp.*, 43(9):645 – 658, 2008.

[23] K. Lee and M. E. O'Sullivan. List decoding of Hermitian codes using Gröbner bases. *J. Symb. Comp.*, 44(12):1662–1675, 2009.

[24] W. Li, J. S. R. Nielsen, and V. R. Sidorenko. On Decoding of Interleaved Chinese Remainder Codes. In *Proc. of MTNS*, 2014. Extended abstract.

[25] M. Mohamed, S. Rizkalla, H. Zoerlein, and M. Bossert. Deterministic Compressed Sensing with Power Decoding for Complex Reed-Solomon Codes. In *Proc. of SCC*, 2015.

[26] T. Mulders and A. Storjohann. On Lattice Reduction for Polynomial Matrices. *J. Symb. Comp.*, 35(4):377–401, 2003.

[27] V. Neiger. Fast computation of shifted Popov forms of polynomial matrices via systems of modular polynomial equations, Feb. 2016.

[28] V. Neiger, J. Rosenkilde, and E. Schost. Fast Computation of the Roots of Polynomials Over the Ring of Power Series. July 2017.

[29] J. Nielsen and P. Beelen. Sub-Quadratic Decoding of One-Point Hermitian Codes. *IEEE Trans. Inf. Theory*, 61(6):3225–3240, June 2015.

[30] J. S. R. Nielsen. Generalised Multi-sequence Shift-Register Synthesis using Module Minimisation. In *Proc. of IEEE ISIT*, pages 882–886, 2013.

[31] J. S. R. Nielsen. *List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2013.

[32] J. S. R. Nielsen. Power Decoding of Reed–Solomon Codes Revisited. In *Proc. of ICMCTA*, Sept. 2014.

[33] J. S. R. Nielsen. Power Decoding of Reed–Solomon Up to the Johnson Radius. In *Proc. of ACCT*, Sept. 2014.

[34] R. R. Nielsen and T. Høholdt. Decoding Reed–Solomon codes beyond half the minimum distance. In *Coding Theory, Cryptography and Related Areas*, pages 221–236. Springer, 1998.

[35] Z. Olesh and A. Storjohann. The vector rational function reconstruction problem. In *Proc. of WWCA*, pages 137–149, 2006.

[36] S. Puchinger and J. R. n. Nielsen. Decoding of interleaved Reed-Solomon codes using improved power decoding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 356–360, June 2017.

[37] J. Rosenkilde and A. Storjohann. Algorithms for Simultaneous Hermite Padé Approximations. In preparation. Extension of [38].

[38] J. Rosenkilde né Nielsen and A. Storjohann. Algorithms for Simultaneous Padé Approximations. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 405–412. ACM, 2016.

[39] R. Roth. *Introduction to Coding Theory*. Cambridge Univ. Press, 2006.

[40] R. Roth and G. Ruckenstein. Efficient Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance. *IEEE Trans. Inf. Theory*, 46(1):246 –257, 2000.

[41] G. Schmidt, V. Sidorenko, and M. Bossert. Decoding Reed-Solomon Codes Beyond Half the Minimum Distance Using Shift-Register Synthesis. In *Proc. of IEEE ISIT*, pages 459–463, 2006.

[42] G. Schmidt, V. Sidorenko, and M. Bossert. Syndrome Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis. *IEEE Trans. Inf. Theory*, 56(10):5245–5252, 2010.

[43] V. Sidorenko and M. Bossert. Fast skew-feedback shift-register synthesis. *Designs, Codes and Cryptography*, 70(1-2):55–67, Jan. 2014.

[44] V. Sidorenko and G. Schmidt. A Linear Algebraic Approach to Multisequence Shift-Register Synthesis. *Problems of Information Transmission*, 47(2):149–165, 2011.

[45] W. A. Stein et al. SageMath Software. http://www.sagemath.org.

[46] A. Storjohann. High-order lifting and integrality certification. *J. Symb. Comp.*, 36(3):613–648, 2003.

[47] M. Sudan. Decoding of Reed–Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997.

[48] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A Method for Solving Key Equation for Decoding Goppa Codes. *Information and Control*, 27(1):87–99, 1975.

[49] P. Trifonov and M. Lee. Efficient Interpolation in the Wu List Decoding Algorithm. *IEEE Trans. Inf. Theory*, 58(9):5963–5971, 2012.

[50] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ. Press, 3rd edition, 2012.

[51] A. Wachter-Zeh, A. Zeh, and M. Bossert. Decoding interleaved Reed–Solomon codes beyond their joint error-correcting capability. *Designs, Codes and Cryptography*, 71(2):261–281, July 2012.

[52] Y. Wu. New List Decoding Algorithms for Reed-Solomon and BCH Codes. *IEEE Trans. Inf. Theory*, 54(8):3611–3630, 2008.

[53] A. Zeh, C. Gentner, and D. Augot. An Interpolation Procedure for List Decoding Reed-Solomon Codes Based on Generalized Key Equations. *IEEE Trans. Inf. Theory*, 57(9):5946–5959, 2011.

[54] A. Zeh, A. Wachter, and M. Bossert. Unambiguous Decoding of Generalized Reed–Solomon Codes Beyond Half the Minimum Distance. In *Proc. of IZS*, 2012.

E-mail address: `jsrn@jsrn.dk`