Power Decoding Reed–Solomon Codes up to the Johnson Radius

Johan S. R. Nielsen jsrn@jsrn.dk
Ulm University, Department of Communications Engineering

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 a tantalising goal to incorporate multiplicities, the parameter allowing the Guruswami–Sudan algorithm to decode up to the Johnson radius. In this paper we show how this can be done, and describe how to efficiently solve the resulting key equations.

1 Introduction

Power decoding was originally proposed by Schmidt, Sidorenko and Bossert for low-rate Reed–Solomon codes (RS) [6]. Using shift-register synthesis techniques, the method allows to decode as many errors as the Sudan algorithm [8]. As opposed to this list decoder, Power decoding returns at most one codeword but will in some cases simply fail. For random errors, this seem to occur with only very small probability, however. So far analytic bounds on the failure probability have been obtained when the powering degree is 2 or 3 [4, 7, 9].

The Sudan decoder generalises to the Guruswami–Sudan decoder [3] by introducing the multiplicity parameter. It seems teasingly clear that one should likewise be able to introduce a "multiplicity parameter" into Power decoding and thereby increase the decoding radius up to the Johnson bound; but such a formulation was so far not found.

In this work we show how it can be done. The overall behaviour of the decoder is similar to Power decoding: 1) the equations are of a generalised shift-register type, and no root-finding as in Guruswami–Sudan is necessary; 2) the decoding radius becomes almost exactly that of the Guruswami–Sudan decoder (under the same choices of parameters); and 3) there remains a low but non-zero probability of failing whenever one decodes beyond half the minimum distance.

In [6, 7], Power decoding was formulated by powering the classical syndrome key equation. In [4] it was described how one can instead power the key equation implicit in Gao's decoder [1]. The Power decoding with multiplicities that we give here is in the same vein.

The results are here stated without proofs but with the necessary lemmas to make those proofs comparatively simple. Furthermore, it should be noted

that the analysis of the algorithm's behaviour is work in progress, and we have mostly conjectured behaviour based on experimental results as of yet.

2 Key Equations

Consider some finite field $\mathbb{F}$. The $[n, k, d]$ Generalised Reed-Solomon (GRS) code is the set

$$\mathcal{C} = \left\{ \big(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)\big) \mid f \in \mathbb{F}[x] \wedge \deg f < k \right\}$$

where $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ are distinct, and the $\beta_1, \ldots, \beta_n \in \mathbb{F}$ are non-zero (not necessarily distinct). The α_i are called *evaluation points* and the β_i *column multipliers*. $\mathcal{C}$ has minimum distance $d = n - k + 1$ and the code is therefore MDS.

Consider now that some $\boldsymbol{c} = (c_1, \ldots, c_n)$ was sent, resulting from evaluating some $f \in \mathbb{F}[x]$, and that $\boldsymbol{r} = (\beta_1 r_1, \ldots, \beta_n r_n) = \boldsymbol{c} + (\beta_1 e_1, \ldots, \beta_n e_n)$ was the received word with (normalised) error $\boldsymbol{e} = (e_1, \ldots, e_n)$. Let $\mathcal{E} = \{i \mid e_i \neq 0\}$ and $\epsilon = |\mathcal{E}|$.

Introduce two essential polynomials, known to the receiver:

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

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

As usual for key equation decoders, the algorithm will revolve around the notion of error locator Λ and error evaluator Ω:

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

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

These four polynomials are related by a key equation, which is implicitly at the heart of Gao's decoder up to half the minimum distance [1]. A slightly stronger relation, however, is the following:

Lemma 1. $\Lambda(f - R) = \Omega G$

From this, it is easy to show the following:

Lemma 2. *For any $s, t \in \mathbb{Z}_+$ and $t \geq s$ then*

$$\Lambda^s (f - R)^t \equiv 0 \mod G^s$$

Combining these two one can then show the main result:

Theorem 1. *For any $s, \ell \in \mathbb{Z}_+$ with $\ell \geq s$, then*

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

Each of these ℓ equations are "key equations" in the following sense: $\Lambda^s, \Lambda^{s-1}\Omega, \ldots, \Lambda\Omega^{s-1}$ are all unknowns, whose inner product with a vector of known polynomials (the $\binom{t}{i}R^{t-i}G^i$) have a remainder modulo G^s of surprisingly low degree (the degree of $\Lambda^s f^t$).

The equations are clearly highly non-linear, so to solve them efficiently, we will relax them to linear equations and hope that the solution to the linear problem turns out to be exactly $\Lambda^s, \ldots, \Lambda\Omega^{s-1}$. This is completely analogous to the approach taken in classical key equation decoding.

3 How to Solve the Key Equations

The linearisation performed to make the solving of the equations of Theorem 1 tractable is the following: we will seek a vector $(\lambda_0, \ldots \lambda_{s-1}, \psi_1, \ldots, \psi_\ell) \in \mathbb{F}[x]^{s+\ell}$ such that the following three requirements are satisfied:

$$\begin{aligned}
\sum_{i=0}^{\min\{s-1,t\}} \lambda_i \cdot \left(\tbinom{t}{i} R^{t-i}G^i\right) &\equiv \psi_t \mod G^s, && \text{for } t = 1, \ldots, \ell \\
\deg \lambda_0 &\geq \deg \lambda_i + i, && \text{for } i = 1, \ldots, s-1 \\
\deg \lambda_0 &\geq \deg \psi_t + t(k-1), && \text{for } t = 1, \ldots, \ell
\end{aligned}$$

Clearly $\mathbf{\Lambda} = (\Lambda^s, \Lambda^{s-1}\Omega, \ldots, \Lambda\Omega^{s-1}, \Lambda^s f, \ldots, \Lambda^s f^\ell)$ satisfies these requirements, but there are unfortunately infinitely many other vectors satisfying them. We will therefore seek the one of least degree, i.e. where $\deg \lambda_0$ is minimal; the hope is then that this vector is $\mathbf{\Lambda}$. In that case, decoding will succeed simply by computing $f = \psi_1/\lambda_0$. If the found minimal vector is *not* $\mathbf{\Lambda}$, then decoding has failed.

The type of key equations of Theorem 1 is dealt with in [5] under the name "2D Padé approximations". The search for a satisfactory and minimal degree vector is there solved using lattice basis reduction of a certain $\mathbb{F}[x]$ matrix. In our case, the matrix would have dimension $(s+\ell) \times (s+\ell)$. The total cost of the lattice basis reduction can be performed with asymptotic complexity $O^\sim(\ell^\omega sn)$, where $O^\sim$ is like big-O but ignoring log-factors in n, s and ℓ; this uses fast matrix multiplication and either the method in [2] or [10]. Alternatively, one can use the Demand–Driven algorithm from [5] which will have asymptotic complexity

$O(\ell s^4 n^2)$[1], but is simple to implement and does not use fast arithmetic methods; therefore it might be more efficient on small and medium sized input.

A central question is how many errors the method will usually cope with. Based on preliminary analysis and experiments, as well as previous results for Power decoding with $s = 1$, we pose the following conjecture:

Conjecture 1. *Let $\tau_{\mathrm{GS}}(s, \ell)$ be the decoding radius of the Guruswami–Sudan algorithm on $\mathcal{C}$ with multiplicity s and list size ℓ. Then decoding using Power decoding, choosing the same s and ℓ will succeed whenever $|\mathcal{E}| \leq \tau_{\mathrm{GS}}(s, \ell) - 1$, with high probability, assuming that all error vectors of the same weight are equi-probable.*

The probability of failing is in the order of $O(q^{-(\tau_{\mathrm{GS}}(s,\ell)-|\mathcal{E}|)})$.

4 Acknowledgements

The author would like to thank Vladimir Sidorenko and Martin Bossert for discussions on this article and Power decoding in general. This work has been supported by the German Research Council "Deutsche Forschungsgemeinschaft" (DFG) under grant BO 867/22-1.

References

[1] 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.

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

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

[4] J. S. R. Nielsen. Power Decoding of Reed–Solomon Codes Revisited. *arXiv*, 1311.1940, Apr. 2014. Submitted to ICMCTA 2014.

[5] J. S. R. Nielsen. Solving generalised Padé approximations over polynomial rings. In *Preprint*, Jan. 2014. Available at http://jsrn.dk/.

[6] 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*, page 459–463, 2006.

[1] If the α_i do not form a multiplicative sub-group of $\mathbb{F}$, then the complexity is instead $O^{\sim}(\ell s^4 n^2)$, and fast polynomial multiplication is used.

[7] 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.

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

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

[10] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symb. Comp.*, 47(7):793–819, July 2012.