Appears in *Proc. 35th Annual Allerton Conference on Communication, Control and Computing*, 1997.

Decoding Reed Solomon Codes beyond the Error-Correction Diameter

Madhu Sudan
Laboratory for Computer Science, MIT
madhu@lcs.mit.edu

Abstract

We describe a new algorithm for the decoding of Reed Solomon codes. This algorithm was originally described in [12]. While the algorithm presented in this article is the same, the presentation is somewhat different. In particular we derive more precise bounds on the performance of the algorithm and show the following: For an $[n, \kappa n, (1-\kappa)n]_q$ Reed Solomon code, the algorithm in [12] corrects $(\epsilon(\kappa)-o(1))n$ errors in polynomial time, where

$$\epsilon(\kappa) = 1 - \frac{1}{1+\rho_\kappa} - \frac{\rho_k}{2}\kappa \text{ where } \rho_\kappa = \left\lfloor \sqrt{\frac{2}{\kappa}+\frac{1}{4}} - \frac{1}{2} \right\rfloor.$$

We also present the following two (hopefully) useful lower bounds on $\epsilon(\kappa)$:

$$\forall \kappa \in [0,1] \quad \epsilon(\kappa) \geq 1 - \sqrt{2\kappa + \frac{\kappa^2}{4}} + \frac{\kappa}{2} \geq 1 - \sqrt{2\kappa}.$$

1 Introduction

For integers n, k and q such that a finite field of size q exists, the Reed Solomon codes are $[n, k, d = n-k]$ codes over the alphabet $F = \mathrm{GF}(q)$ (the Galois field of order q). The code may be obtained by letting a message $m = m_0 \ldots m_{k-1}$ denote the coefficients of a degree $k-1$ polynomial $M(x) = \sum_{j=0}^{k-1} m_j x^j$ and letting the encoding of m be $C(m) = c_1 \ldots c_n$ where $c_i = M(x_i)$ where $x_1, \ldots, x_n$ are n distinct elements of F. (It is standard to pick $n = q-1$, though not necessary to prove the distance property or for the algorithms we describe).

In this paper we consider the following bounded distance decoding problem:

`Input`: Integers n, k and e; and n pairs $\{(x_i, y_i)\}_{i=1}^n$, $x_i, y_i \in F$ with x_i's being distinct[1].
`Goal`: Find all polynomials $p_1, \ldots, p_m$ of degree $k-1$ such that for every $j \in [m]$, $p_j(x_i) \neq y_i$ for at most e values of $i \in [n]$.

As can be seen easily, this problem captures the bounded distance decoding problem for Reed Solomon codes. There is a rich history of work associated with this problem. The classical work of Berlekamp-Massey (cf. [2, 9]), corrects upto $\lfloor (d-1)/2 \rfloor$ errors. Sidelnikov [11] and Dumer [4] have constructed algorithms which correct up to $\lfloor (d-1)/2 \rfloor + c \log n$ errors for any constant c [4, 11]. We give an algorithm that improves over these results when k/n is sufficiently small (i.e., less than 1/3). Our algorithm is motivated by an algorithm of Welch and Berlekamp [14, 3] which corrects $\lfloor (d-1)/2 \rfloor + 1$ errors. In this article we describe the algorithm of Welch and Berlekamp and use it motivate our decoding algorithm. We also describe a crucial intermediate step from [1] which forms the basis of our algorithm. Our main result is summarized below and proven in Lemma 9.

Theorem 1 *For every ϵ, κ The bounded distance decoding problem with parameters n, $k = \kappa n$ and $e = \epsilon(\kappa) n$ can be solved in polynomial time provided*

$$\epsilon(\kappa) < 1 - \frac{1}{1+\rho_\kappa} - \frac{\rho_\kappa}{2}\kappa \text{ where } \rho_\kappa = \left\lfloor \sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{1}{2} \right\rfloor.$$

2 Decoding with univariate rational functions [14, 3]

The idea of Welch and Berlekamp [14, 3] can be informally described as follows: They describe how a "rational function" in x can be used to "explain" the "data" $\{(x_i, y_i)\}_{i=1}^n$. They then show how to efficiently *find* a rational function that explains the data and then show how to use this rational function to find the (unique) polynomial p which disagrees with the data in at most e places. We now describe the algorithms more formally.

Lemma 2 ([14, 3]) *Given n points $\{(x_i, y_i)\}_{i=1}^n$ such that there exists a degree $k-1$ polynomial p such that $y_i \neq p(x_i)$ for at most e values of i, the following hold:*

1. *There exist polynomials $N(x)$ and $D(x)$ where $\deg(N) \leq k-1+e$, D is monic with $\deg(D) = e$, such that for every $i \in [n]$, $N(x_i) = y_i D(x_i)$. (Informally, we could say that y_i equals the rational function $\frac{N}{D}(x_i)$.)*
2. *Such a pair of polynomials (N, D) can be found in polynomial time.*
3. *For any such pair of polynomials (N, D), $\frac{N}{D}(\cdot) = p(\cdot)$, provided $e \leq \frac{1}{2}(n-k)$.*

Remark: As a consequence p can be found in polynomial time. We just divide the polynomials N and D to obtain p.

[1] For our algorithm from [12] we can replace this condition with the weaker one that (x_i, y_i)'s are distinct.

Proof: Let $E(x)$ be an "error-locator" polynomial, i.e., $E(x_i) = 0$ if (but not necessarily only if) $y_i \neq p(x_i)$. Notice that E has degree at most e and w.l.o.g. we can allow it to be monic and have degree exactly e. We now notice that the polynomials $D(\cdot) = E(\cdot)$ and $N(\cdot) = p(\cdot)E(\cdot)$ satisfy the condition $N(x_i) = y_i D(x_i)$ for every i. This proves Part 1.

To see Part 2, notice that if we let the unknowns $\{n_j\}_{j=0}^{e+k-1}$ denote the coefficients of $N(\cdot)$ and let the unknowns $\{d_j\}_{j=0}^{e}$ denote the coefficients of $D(\cdot)$, then the constraints $N(x_i) = y_i D(x_i)$ give linear constraints on the unknowns $\{n_j\}$'s and $\{d_j\}$'s. Also the constraint $\deg(D) = e$. We do so by setting the linear constraint $d_e = 1$. Thus a solution pair (N, D) can be found efficiently by solving a linear system.[2]

Finally notice that there exists a pair (N, D) such that $\frac{N}{D}(\cdot) = p(\cdot)$, as described in the proof of Part 1. Thus to prove Part 3, it suffices to prove that for any pair of solutions (N_1, D_1) and (N_2, D_2), satisfying $N_j(x_i) = y_i D_j(x_i)$ every $i \in [n]$ and $j \in \{1, 2\}$, $\frac{N_1}{D_1}(\cdot) = \frac{N_2}{D_2}(\cdot)$. To see this, first observe that for every $i \in [n]$, we have $N_1(x_i)D_2(x_i)y_i = N_2(x_i)D_1(x_1)y_i)$. Futhermore, we can cancel y_i from both sides (even if $y_i = 0$, since in such a case we have $N_1(x_i) = N_2(x_i) = 0$. This yields that for every i, $(N_1D_2)(x_i) = (N_2D_1)(x_i)$. But then both sides describe polynomials of degree $2e + k - 1$ and two sides agree on n points. By the condition on n, we have that $n > 2e + k - 1$ and thus the polynomials on the two sides are identical, i.e., $(N_1D_2)(\cdot) = (N_2D_1)(\cdot)$. This yields the desired conclusion immediately. ▌

3 Decoding with algebraic curves in the plane [1, 12]

A slightly different interpretation of the Welch-Berlekamp algorithm is that it finds an algebraic curve in the plane which "explains" the "data". To be precise, the algorithm finds a function $Q(x, y)$, where $Q(x, y) = D(x)y - N(x)$, such that for every $i \in [n]$, $Q(x_i, y_i) = 0$. While this particular scenario attempts to explain the data by a "linear" polynomial in y - there is no need to restrict the analysis to this situation. Ar et al. [1] considered such a generalization. They consider the case where the data is "explained" some algebraic curve Q of low degree in y (but not necessarily a linear polynomial in y). They show that in such a case, if there exists a polynomial p such that $y_i = p(x_i)$ for many values of i (compared to $\deg_x(Q)$ and $\deg_y(Q)$) then p can be reconstructed easily. To describe their analysis, the following definition is useful.

Definition 3 *For positive integers w_x and w_y, the (w_x, w_y)-weighted degree of a bivariate polynomial $Q(x, y) = \sum_{i,j} q_{ij}x^i y^j$ is defined to be* $\max\{iw_x + jw_y | q_{ij} \neq 0\}$.

Lemma 4 ([1]) *Given n points $\{(x_i, y_i)\}_{i=1}^n$ s.t. there exists a bivariate polynomial Q satisfying:*

$$\text{The } (1, k-1) \text{ weighted degree of } Q \text{ is at most } D,\ Q \not\equiv 0 \text{ and } \forall i \in [n],\ Q(x_i, y_i) = 0. \quad (1)$$

Then the following hold:

[2] Actually Berlekamp and Welch [14, 3] give a much more efficient solution for this task, but we will not describe their solution here.

1. *A polynomial Q satisfying equation (1) can be found in polynomial time.*

2. *If p is a polynomial in x of degree at most $k-1$ such that $y_i \neq p(x_i)$ for at most $e < n - D$ values of i then for any polynomial Q satisfying (1), it is the case that $(y - p(x))$ divides $Q(x, y)$.*

Remark: As a consequence, a small set of polynomials which includes p can be found in polynomial time. We simply factor the polynomial Q obtained in Part 1 above and output all p such that $y - p(x)$ divides Q. The polynomial Q can be factored in time polynomial in its degree using the algorithm of Kaltofen [7] or Grigoriev [6] (see also Kaltofen [8]).

Proof: For Part 1, we observe as in the proof of Lemma 2 that for any i, the condition $Q(x_i, y_i) = \sum_{jl} q_{jl} x^j y^l = 0$ is a linear constraint on the unknowns q_{jl}. Thus a solution satisfying (1) can be found in polynomial time, if one exists.

For Part 2, we consider the polynomial $g(x) \stackrel{\text{def}}{=} Q(x, p(x))$. Notice that since the $(1, k-1)$-weighted degree of Q is D, the degree of g is also at most D. Notice further that if for some $i \in [n]$, $y_i = p(x_i)$, then $g(x_i) = Q(x_i, p(x_i)) = Q(x_i, y_i) = 0$. Thus g is zero on $n - e > D$ points. Thus g is identically 0. Now consider the polynomial $Q_x(y) \stackrel{\text{def}}{=} Q(x, y)$ (i.e., the polynomial Q viewed as a polynomial in y with coefficients from the ring of polynomials in x). By the division theorem for polynomials we have that if $Q_x(\zeta) = 0$, then $y - \zeta$ divides $Q_x(y)$. Applying this fact to $Q_x(p(x)) = Q(x, p(x)) = g(x) = 0$ we find that $y - p(x)$ divides $Q(x, y)$. ▮

Notice the close correspondence between Lemmas 2 and 4. The main difference between the two is Part 1 of Lemma 2, the analog of which is missing in Lemma 4. As a result Lemma 4 works conditionally, i.e., only when the "data" is explained by a low-degree algebraic curve. Our solution complements this by providing a "triviality" result. Namely, we observe that every set of points lies on a low degree algebraic curve; low enough to make Lemma 4 always useful! We illustrate this observation by a simple example

Example 5 *For any set of points $\{x_i, y_i\}_{i=1}^n$ there exists a non-zero polynomial Q with* $\deg_x(Q), \deg_y(Q) \leq \lceil \sqrt{n} \rceil$.

Proof: Consider the linear system $Q(x, y) = \sum_{j=0}^{\lceil \sqrt{n} \rceil} \sum_{l=0}^{\lceil \sqrt{n} \rceil} q_{jl} x_i^j y_i^l = 0$. For every i this forms a homogenous linear system. The system has n constraints on $(\lceil \sqrt{n} + 1 \rceil)^2 \geq (\sqrt{n} + 1)^2 \geq n + 1$ unknowns. By just counting the number of unknowns we know that the system has a non-zero solution (any homogenous linear system with more variables than unknowns has a non-zero solution). ▮

Notice that unlike in Part 1 of Lemma 2, we didn't even use the fact that the data has some agreement with a degree $k-1$ polynomial. Thus the example above is a "triviality" result - it holds for any set of n points. Yet the trivial result can be useful as seen next:

Example 6 *For any set of n distinct points $\{(x_i, y_i)\}_{i=1}^n$ the following hold:*

1. *There exists a bivariate polynomial Q satisfying equation (1) with $D = k\sqrt{n}$.*

2. *Such a polynomial Q can be found in polynomial time.*

3. *If p is a polynomial in x of degree at most $k-1$ such that $y_i \neq p(x_i)$ for at most $e < n - k\lceil\sqrt{n}\rceil$ values of i then for any polynomial Q satisfying (1) with $D = k\sqrt{n}$, it is the case that $(y - p(x))$ divides $Q(x, y)$.*

Proof: Part 1 follows from Example 5. Parts 2 and 3 follow from Parts 1 and 2 of Lemma 4 with $D = k\lceil\sqrt{n}\rceil$. ∎

Notice that Example 6 is already correcting more errors than guaranteed by Lemma 2 for some values of n and k (in particular when k grows as $o(\sqrt{n})$. By some fine tuning we can actually get to the point where this algorithm always does at least as well as the Welch-Berlekamp algorithm and for rate less than 1/3 it starts correcting significantly more error. The fine tuning is performed by minimizing the weighted $(1, k-1)$ degree of the polynomial Q which is used to explain the data.

Lemma 7 *Given n and k, let t be the smallest positive integer satisfying:*

$$(t-1)\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor+1\right)-\frac{(k-1)}{2}\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor+1\right)\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor\right)>n.$$

Then, for any set of n distinct points $\{(x_i, y_i)\}_{i=1}^n$ the following hold:

1. *There exists a bivariate polynomial Q satisfying:*

$$\text{The } (1, k-1) \text{ weighted degree of } Q \text{ is } \leq t,\ Q \not\equiv 0 \text{ and } \forall i \in [n],\ Q(x_i, y_i) = 0. \tag{2}$$

2. *A polynomial Q satisfying (2) can be found in polynomial time.*

3. *If p is a polynomial in x of degree at most $k-1$ such that $y_i \neq p(x_i)$ for at most $e < n-t$ values of i then for any polynomial Q satisfying (2), it is the case that $(y-p(x))$ divides $Q(x, y)$.*

Remark: Again as a consequence we can find a list of all polynomials p that agree with the points $\{(x_i, y_i)\}$ in $n - e$ points in time polynomial in n.

Proof: A polynomial Q of $(1, k-1)$-weighted degree t is allowed to have a non-zero coefficient q_{jl} if $j + (k-1)l \leq t$. Counting the number of such coefficients we find that Q has

$$\sum_{l=0}^{\lfloor\frac{t-1}{k-1}\rfloor}(t-(k-1)l+1)=(t-1)\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor+1\right)-\frac{(k-1)}{2}\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor+1\right)\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor\right)$$

coefficients. Since this number is strictly greater than n we must have a non-zero solution. This proves Part 1. Parts 2 and 3 follow directly from Lemma 4. ∎

4 Some bounds

Let $e(k,n)$ denote the maximum number of errors corrected by the algorithm of Lemma 7 on an $[n,k,d]_q$ Reed Solomon code. Let $\kappa = k/n$ denote the rate of a code and let $\epsilon(\kappa) \triangleq \lim_{n\to\infty} e(\kappa n, n)/n$ denote the asymptotic error-correcting rate of the algorithm. In this section we derive the exact form of $e(k,n)$ and $\epsilon(\kappa)$. We also describe some lower bounds to both quantities.

Lemma 8

$$e(k,n) = n-(k-1)r_{k,n} - \left\lceil \frac{2n-(k-1)r_{k,n}(r_{k,n}+1)}{2(r_{k,n}+1)} \right\rceil - 2 \; where \; r_{k,n} = \left\lfloor \sqrt{\frac{2(n+1)}{k-1}+\frac{1}{4}} - \frac{1}{2} \right\rfloor.$$

Proof: Let t be the smallest integer satisfying

$$(t-1)\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor+1\right) - \frac{(k-1)}{2}\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor+1\right)\left(\left\lfloor\frac{t-1}{k-1}\right\rfloor\right) > n.$$

By Lemma 7 $e(k,n) = n-t-1$. Let $t-1 = r(k-1)+s$, where r and s are integers and $0 \le s < k-1$. Then $\lfloor \frac{t-1}{k-1} \rfloor = r$; and r and s satisfy

$$(r(k-1)+s)(r+1) - \frac{(k-1)(r+1)r}{2} > n. \tag{3}$$

We first determine r based on the above. The constraints $0 \le s < k-1$ translate into the constraints:

$$r(k-1)(r+1) - r(k-1)(r+1)/2 \le n+1 \text{ and } (r+1)(k-1)(r+1) - r(k-1)(r+1)/2 > n+1.$$

The above, in turn, simplify to

$$r \le \sqrt{\frac{2(n+1)}{k-1} - \frac{1}{4}} - \frac{1}{2} \text{ and } r > \sqrt{\frac{2(n+1)}{k-1} - 1} - 1.$$

The requirement that r be an integer now allows us to determine r precisely:

$$r = \left\lfloor \sqrt{\frac{2(n+1)}{k-1}+\frac{1}{4}} - \frac{1}{2} \right\rfloor.$$

The value s can now be determined from (3) and we get:

$$s = \left\lceil \frac{2(n+1) - r(r+1)(k-1)}{2(r+1)} \right\rceil.$$

The lemma follows by setting $r_{k,n} = r$ and $t = r(k-1)+s+1$ and using the fact that $e(k,n) = n-t-1$. ▮

The following lemma simplifies some of the expressions above by examining the error-correction rate $\epsilon(\kappa)$ as a function of κ.

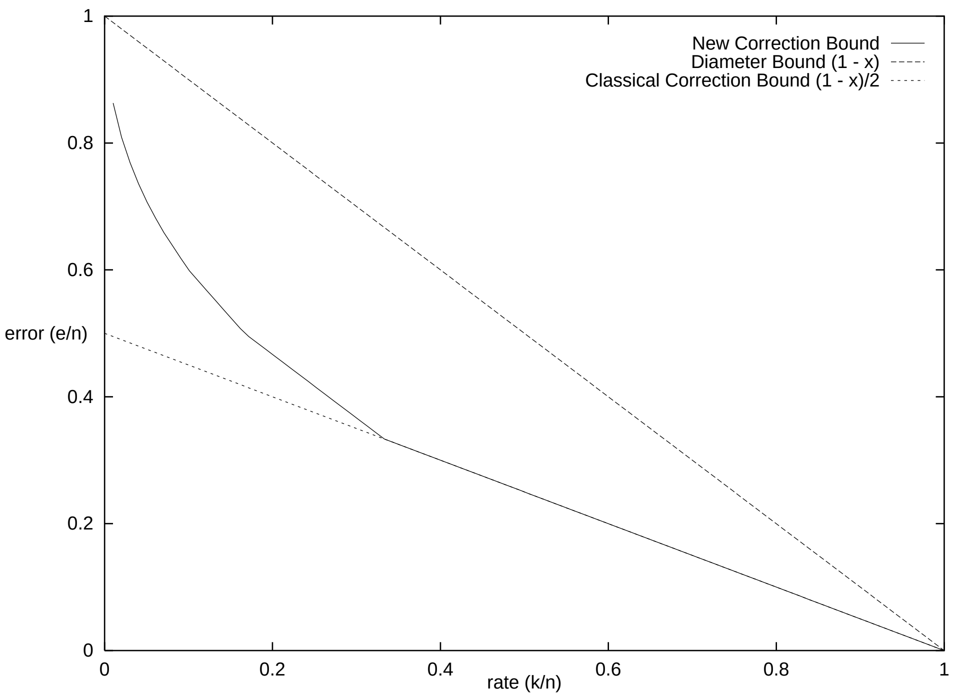

Figure 1: The new error correction bound $\epsilon(\kappa)$ plotted against the rate κ. Also described are the error correction radius of the code and the distance of the code.

Lemma 9

$$\epsilon(\kappa) = 1 - \frac{1}{1+\rho_\kappa} - \frac{\rho_\kappa}{2}\kappa \text{ where } \rho_\kappa = \left\lfloor \sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{1}{2} \right\rfloor. \tag{4}$$

$\epsilon(\kappa)$ can be lower bounded as follows:

$$\forall \kappa \in [0,1], \quad \epsilon(\kappa) \geq 1 - \sqrt{2\kappa + \frac{\kappa^2}{4}} + \kappa/2 \tag{5}$$

$$\geq 1 - \sqrt{2\kappa} \tag{6}$$

Remark: The bound (4) is described pictorially in Figure 1. The lower bounds (5) and (6) are compared against the bound (4) in Figure 2.

Proof: (4) follows from Lemma 8 by letting $\rho_\kappa = \lim_{n\to\infty} r_{\kappa n,n}$ and simplifying the quantity $\lim_{n\to\infty} e(\kappa n, n)/n$.

To prove (5), we first show that

$$1 - \frac{1}{1+\rho_\kappa} - \frac{\rho_\kappa}{2}\kappa \geq 1 - \frac{1}{1+\rho'_\kappa} - \frac{\rho'_\kappa}{2}\kappa, \tag{7}$$

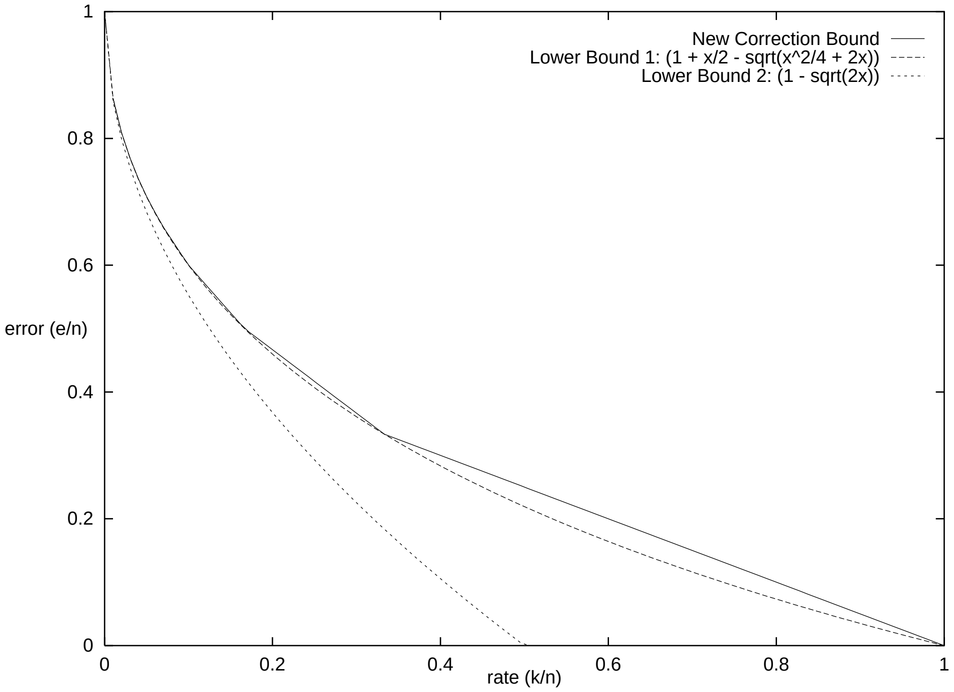

Figure 2: Lower bounds on the error-correction rate of our algorithm.

where $\rho'_\kappa = \sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{3}{2}$. Notice that by the definition of ρ'_κ, we have

$$\rho'_\kappa \le \rho_\kappa \le \rho'_\kappa + 1 \quad \text{and} \quad \frac{2}{\kappa} = (\rho'_\kappa + 2)(\rho'_\kappa + 1).$$

(7) follows from:

$$\begin{array}{rrcll} & \frac{1}{1+\rho'_\kappa} - \frac{1}{1+\rho_\kappa} + \frac{\rho'_\kappa}{2}\kappa - \frac{\rho_\kappa}{2}\kappa & \ge & 0 & \\ \Leftarrow & (\rho_\kappa - \rho'_\kappa)\left(\frac{1}{(1+\rho'_\kappa)(1+\rho_\kappa)} - \frac{\kappa}{2}\right) & \ge & 0 & \\ \Leftarrow & \frac{1}{(1+\rho'_\kappa)(1+\rho_\kappa)} - \frac{\kappa}{2} & \ge & 0 & (\text{Since } \rho_\kappa \ge \rho'_\kappa) \\ \Leftarrow & \frac{1}{(1+\rho'_\kappa)(1+\rho_\kappa)} & \ge & \frac{1}{(1+\rho'_\kappa)(2+\rho'_\kappa)} & (\text{Since } \frac{2}{\kappa} = (1+\rho'_\kappa)(2+\rho'_\kappa)) \\ \Leftarrow & 2 + \rho'_\kappa & \ge & 1 + \rho_\kappa. & \end{array}$$

where the last inequality follows from the properties of ρ'_κ.

(5) now follows from the following equalities:

$$\begin{aligned} & 1 - \frac{1}{1 + \sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{3}{2}} - \frac{\sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{3}{2}}{2}\kappa \\ = \; & 1 - \frac{1}{\sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{1}{2}} - \frac{\sqrt{\frac{2}{\kappa} + \frac{1}{4}} - \frac{3}{2}}{2}\kappa \end{aligned}$$

$$
\begin{aligned}
&= 1 - \frac{\sqrt{\frac{2}{\kappa}+\frac{1}{4}}+\frac{1}{2}}{\frac{2}{\kappa}} - \frac{\sqrt{\frac{2}{\kappa}+\frac{1}{4}}-\frac{3}{2}}{2}\kappa \\
&= 1 - \sqrt{2\kappa+\frac{\kappa^2}{4}}+\frac{\kappa}{2}.
\end{aligned}
$$

Finally inequality (6) is derived easily as follows:

$$
\begin{aligned}
1+\frac{\kappa}{2}-\sqrt{2\kappa+\frac{\kappa^2}{4}} &\geq 1-\sqrt{2\kappa} \\
\Leftarrow \sqrt{2\kappa+\frac{\kappa^2}{4}} &\leq \sqrt{2\kappa}+\frac{\kappa}{2} \\
\Leftarrow 2\kappa+\frac{\kappa^2}{4} &\leq 2\kappa+\frac{\kappa^2}{4}+\kappa\sqrt{2\kappa}
\end{aligned}
$$

$\blacksquare$

5 Conclusions

The algorithm as presented here does generalize naturally to dealing with erasures, for the simple reason that we made no assumptions about the x_i's (we didn't depend on them being in some specific order etc.). However the bounds can not be represented elegantly in any form - so we do not attempt to do so here. We only point out that since the Welch-Berlekamp algorithm is a special case of ours, we do at least as well.

There are a number of questions on decoding Reed Solomon codes that remain open. For instance, is there an algorithm that can decode from more errors (than $(d-1)/2$) when $\kappa = k/n > 1/3$? A nice target would be a decoding algorithm that works for $\epsilon(\kappa) \leq 1-\sqrt{\kappa}$. In this case we know (cf. [5, 10]) that the number of codewords within a distance of $\epsilon(\kappa)n$ is bounded by a polynomial in n. One does expect that the problem will become harder as $\epsilon(\kappa) \to 1-\kappa$. It would be interesting to see if the problem becomes NP-hard as $\epsilon(\kappa) \to 1-\kappa$.

Acknowledgments

I am grateful to Dave Forney, Simon Litsyn, Ronny Roth, and Alex Vardy for their valuable comments on the article [12]. Their detailed comments motivated and encouraged the writing of this article.

References

[1] S. Ar, R. Lipton, R. Rubinfeld and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, to appear. Preliminary version in

Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, pp. 503–512, 1992.

[2] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw Hill, New York, 1968.

[3] E. R. Berlekamp. Bounded Distance +1 Soft-Decision Reed-Solomon Decoding. In *IEEE Transactions on Information Theory*, pages 704-720, vol. 42, no. 3, May 1996.

[4] I. I. Dumer. Two algorithms for the decoding of linear codes. *Problems of Information Transmission*, 25(1):24–32, 1989.

[5] O. Goldreich, R. Rubinfeld and M. Sudan. Learning polynomials with queries: The highly noisy case. *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pp. 294–303, 1995.

[6] D. Grigoriev. Factorization of Polynomials over a Finite Field and the Solution of Systems of Algebraic Equations. Translated from Zapiski Nauchnykh Seminarov Lenningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR, Vol. 137, pp. 20-79, 1984.

[7] E. Kaltofen. A Polynomial-Time Reduction from Bivariate to Univariate Integral Polynomial Factorization. In *23rd Annual Symposium on Foundations of Computer Science*, pages 57-64, 1982.

[8] E. Kaltofen. Polynomial factorization 1987–1991. *LATIN '92*, I. Simon (Ed.) Springer LNCS, v. 583:294–313, 1992.

[9] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1981.

[10] J. Radhakrishnan. *Personal communication*, January, 1996.

[11] V. M. Sidelnikov. Decoding Reed Solomon codes beyond $(d-1)/2$ and zeros of multivariate polynomials. *Problems of Information Transmission*, 30(1):44–59, 1994.

[12] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180-193, March 1997.

[13] A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pp. 92–109, 1997.

[14] L. R. Welch and E. R. Berlekamp. Error correction of algebraic block codes. *US Patent* Number 4,633,470, issued December 1986.