Lecture 6
Reed-Solomon Codes

In this class we will study a very important class of evaluation codes: The Reed-Solomon codes. Reed-Solomon codes are being used as outer codes in many communication systems. Many of today's ubiquitous devices such as disk drives or CD/DVD players rely heavily on the error correction capabilities of Reed-Solomon codes. This class introduces the basic concepts, and some abstract decoding algorithms. Later lectures will focus on efficient implementations of these algorithms.

6.1. Reed-Solomon Codes as Evaluation Codes

Let $\mathbb{F}_q[x]_{<k}$ denote the $\mathbb{F}_q$-space of univariate polynomials of degree less than k over $\mathbb{F}_q$. Further, let $S = \{x_1, \ldots, x_n\}$ be a subset of $\mathbb{F}_q$. We consider the evaluation code with parameters $(\mathbb{F}_q[x]_{<k}, S, \mathbb{F}_q)$, where the action of $\mathbb{F}_q[x]_{<k}$ on S is given by polynomial evaluation. In other words, the code we are considering is the image of the homomorphism:

$$\begin{array}{rcl} \mathbb{F}_q[x]_{<k} & \to & \mathbb{F}_q^n, \\ f & \mapsto & (f(x_1), \ldots, f(x_n)). \end{array} \tag{6.1}$$

We call this image a *Reed-Solomon code* (or RS code), after the two inventors *Irving Reed* and *Gus Solomon* of these codes. If we want to be very precise, then we denote this code by $\mathrm{RS}(k; x_1, \ldots, x_n)$.

Why should this code be good? The reason is, that we can control the number of zeros in a codeword, using the following well-known theorem.

Theorem 6.1. *A nonzero polynomial f of degree r over a field $\mathbb{F}$ has at most r roots in $\mathbb{F}$.*

This means that if $k < n$, then the image of a nonzero polynomial $f \in \mathbb{F}_q[x]_{<k}$ has at most $k-1$ zero coordinates, and hence the minimum distance of the corresponding code is at least $n - k + 1$. Moreover, by the same token, no nonzero polynomial can be mapped to the zero codeword, hence the mapping is injective, hence the dimension of the code is k. Since by the Singleton inequality the minimum distance cannot be larger than $n - k + 1$, we see that the minimum distance is equal to $n - k + 1$.

Theorem 6.2. *If $k < n$, then the code $\mathrm{RS}(k; x_1, \ldots, x_n)$ has dimension k and minimum distance $n-k+1$. The code is thus MDS.*

It is clear that the following matrix *Vandermonde* matrix is a generator matrix for $\mathrm{RS}(k; x_1, \ldots, x_n)$:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{k-1} & x_2^{k-1} & x_3^{k-1} & \cdots & x_n^{k-1} \end{pmatrix}.$$

RS-codes, as described here, are non-systematic. They can be made systematic in a variety of ways, some of which we will describe when we study the displacement method.

Lecture and scribes by Amin Shokrollahi

6.2. The Welch-Berlekamp Algorithm

The decoding problem is the following. We are given a word $y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ with a promise that it has distance $\leq (n-k)/2$ from a q-ary RS-code $C = \mathrm{RS}(k; x_1, \ldots, x_n)$. The task is to find the closest codeword in C to y. We call this codeword c, and call f the corresponding polynomial in $\mathbb{F}_q[x]_{<k}$. In other words, for all i, $c_i = f(x_i)$.

We use the concept of an error locator described in the previous section. Let us call a polynomial $h(x)$ an *error locator* if $h(x_i) = 0$ if and only if the vector y and its closest codeword c differ in the i-th position. Finding the error locator is tantamount to decoding, since we can discard the positions in error, and perform an erasure decoding on the rest of the vector y.

What do we know about $h(x)$? We know that its degree is at most $(n-k)/2$, and that

$$\forall\, i = 1, \ldots, n: \quad (f(x_i) - y_i) \cdot h(x_i) = 0.$$

or stated differently

$$\forall\, i = 1, \ldots, n: \quad g(x_i) - y_i h(x_i) = 0, \tag{6.2}$$

where $g(x) := f(x)h(x)$ is a polynomial of degree $< k + (n-k)/2 = (n+k)/2$. It does not seem like this equation can give us anything useful, since we only know the y_i. However, it turns out that *any* nontrivial solution to (6.2) with $\deg(h) \leq (n-k)/2$ and $\deg(g) < (n+k)/2$ will solve the decoding problem, i.e., for any nonzero pair (g, h) with the given degree constraints we have $f = g/h$!

Theorem 6.3. *Suppose that $(g, h) \in \mathbb{F}_q[x]_{<(n+k)/2} \times \mathbb{F}_q[x]_{\leq (n-k)/2}$ is a nonzero pair of polynomials satisfying* (6.2)*. Then $h(x)$ is an error locator, g is divisible by h, and $f(x) = g(x)/h(x)$.*

Proof. Consider the polynomial $F(x) := g(x) - f(x)h(x) \in \mathbb{F}_q[x]_{<(n+k)/2}$. Note that for all i we have

$$F(x_i) = g(x_i) - f(x_i)h(x_i) = h(x_i)(y_i - f(x_i)).$$

So, if i is such that $y_i = f(x_i)$, or in other words, if i is not in error, then $F(x_i) = 0$. Since there are at most $(n-k)/2$ error positions, there are at least $(n+k)/2$ positions not in error, so F has at least $(n+k)/2$ zeros. But its degree is less than $(n+k)/2$. So, $F(x)$ is identically zero, and hence $g(x) = f(x)h(x)$. The fact that $h(x)$ is an error locator follows from the above displayed equation. □

Does a nonzero solution to (6.2) exist, and if so, how can we calculate it? Suppose that $g(x) = g_0 + g_1 x + \cdots + g_{(n+k)/2-1} x^{(n+k)/2-1}$, and $h(x) = h_0 + h_1 x + \cdots + h_{(n-k)/2} x^{(n-k)/2}$, with unknown coefficients g_i and h_i. Then (6.2) translates to the following system of linear equations:

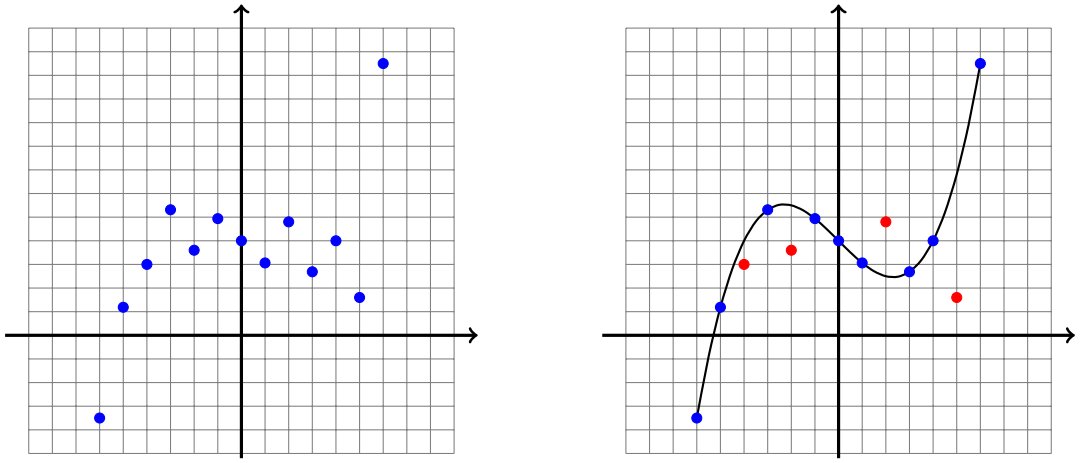
$$\left(\begin{array}{cccc|cccc} 1 & x_1 & \cdots & x_1^{(n+k)/2-1} & -y_1 & -y_1x_1 & \cdots & -y_1x_1^{(n-k)/2} \\ 1 & x_2 & \cdots & x_2^{(n+k)/2-1} & -y_1 & -y_2x_2 & \cdots & -y_2x_2^{(n-k)/2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{(n+k)/2-1} & -y_n & -y_nx_n & \cdots & -y_nx_n^{(n-k)/2} \end{array}\right) \cdot \left(\begin{array}{c} g_0 \\ g_1 \\ \vdots \\ g_{(n+k)/2-1} \\ \hline h_0 \\ h_1 \\ \vdots \\ h_{(n-k)/2} \end{array}\right) = 0. \tag{6.3}$$

This is a homogeneous system of n equations in $(n+k)/2 + (n-k)/2 + 1 = n+1$ unknowns, so it has a nonzero solution. Moreover, such a solution can be found by solving this system using, for example, the Gaussian elimination algorithm. However, since the matrix corresponding to this system is *structured*, there are much faster methods for its solution. We will review such methods later in the context of the displacement method.

Summarizing, the Welch-Berlekamp (WB) decoder can be described as follows:

1. Find a nonzero solution of the system (6.3) and form the polynomials $g(x)$ and $h(x)$.
2. Compute $f(x) = g(x)/h(x)$.
3. The closest codeword to the received word is $(f(x_1), \ldots, f(x_n))$.

The WB-decoder has an interesting geometric interpretation. Suppose that we are given n pairs (x_i, y_i) where the x_i are different, and we are asked to fit a polynomial of degree $< k$ through these points so that it passes through as many of the given points as possible. Then, as long as at least $(n+k)/2$ of the points are "correct," i.e., lie on the same polynomial of degree $< k$, we can find the polynomial using the WB-decoder. The situation is exemplified in Figure 6.1 for the case $n = 13$, and $k = 4$.



(a) (b)

Figure 6.1: (a) We are given 13 points of which we know that at least 9 lie on a third degree polynomial; (b) a plot of the unique polynomial passing through 9 of the points. It reveals the four outliers.

6.3. Decoding More Errors: List Decoding

We know that for a given RS-code with parameters $[n, k, n-k+1]_q$, we can uniquely decode up to $e = (n-k)/2$ errors. What happens, though, when the number of errors is larger? In that case we cannot possibly hope for unique decoding in general, but is it possible to at least find all codewords that are with a given Hamming-radius of the received word?

Given the received vector y, and a real number τ, we would like to find all codewords of distance $\leq \tau n$ from y. Phrased this way, the WB-decoder solves the problem when $\tau = (1-R)/2$, $R = k/n$ being the rate of the code. Can we do better?

Let us change our view a little, and consider the received vector as a set of points $(x_i, y_i) \in \mathbb{F}_q^2$ with pairwise distinct x-coordinates. Our task would be to find all polynomials $f(x) \in \mathbb{F}_q[x]_{<k}$ passing through at least $n - \tau n$ of the points.

Suppose that there are ℓ such polynomials, $f_1(x), \ldots, f_\ell(x)$. Then all the points (x_i, y_i) would be zeros of

$$Q(x, y) := (y - f_1(x)) \cdot (y - f_2(x)) \cdots (y - f_\ell(x)) \cdot E(x),$$

with a polynomial $E(x)$ of degree $< e$, which is a summary for all the points that are in error for all the f_i's. If we had $Q(x, y)$ from somewhere, then we could find the polynomials $f_1(x), \ldots, f_\ell(x)$ through finding all factors of the form $y - g(x)$ of $Q(x, y)$.

This is the Ansatz we are going to use: finding a nonzero polynomial $Q(x, y) = Q_\ell(x)y^\ell + Q_{\ell-1}(x)y^{\ell-1} + \cdots + Q_1(x)y + Q_0(x)$, with polynomials $Q_i(x)$ of degree $< e + (\ell - i)(k-1)$ such that

$$\forall\, i = 1, \ldots, n: \qquad Q(x_i, y_i) = 0. \tag{6.4}$$

(We have replaced $E(x)$ with $Q_\ell(x)$ for consistency reasons.) Under certain conditions on τ, k, and n, it turns out that *any* such polynomial Q will have the property that $Q(x, f(x)) = 0$ if $y - f(x)$ passes through at least $n - \tau n$ of the points (x_i, y_i). This idea for doing a *list decoding* is due to M. Sudan, and is known as the Sudan list-decoding algorithm.

Theorem 6.4. *Suppose that $Q(x, y) = Q_\ell(x)y^\ell + \sum_{i=0}^{\ell-1} Q_i(x)y^i$ with polynomials $Q_i(x)$ of degree less than $e + (\ell - i)(k-1)$, and that $Q(x, y)$ satisfies the equations in (6.4). Then any polynomial $f(x) \in \mathbb{F}_q[x]_{<k}$ for which $(f(x_1), \ldots, f(x_n))$ has a distance of at most $n - e - \ell(k-1)$ of the received vector y satisfies $Q(x, f(x)) = 0$.*

Proof. Suppose that $f(x)$ is such that $c = (f(x_1), \ldots, f(x_n))$ has a distance $\leq n - e - \ell(k-1)$ from the received word. Consider the polynomial $F(x) := Q(x, f(x))$. Then the degree of $F(x)$ is less than $e + \ell(k-1)$. Moreover,

$$c_i = y_i \implies F(x_i) = 0.$$

This means that $F(x)$ vanishes at all positions i at which c and y agree. By assumption, c and y agree on at least $e + \ell(k-1)$ positions. But the degree of $F(x)$ is less than $e + \ell(k-1)$, which shows that $F(x) = 0$. □

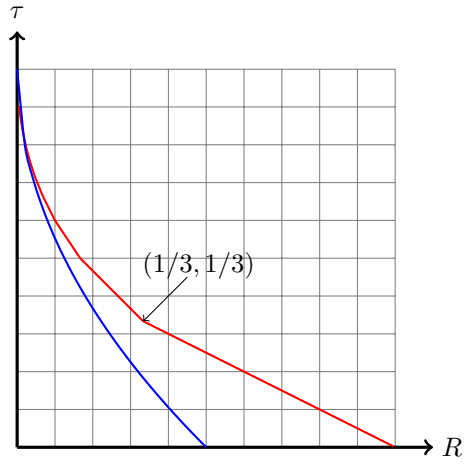


Figure 6.2: The upper curve is the relative error correction capability of Sudan's algorithm versus the rate of the code. It consists of pairwise linear functions. The lower curve is the plot of $1 - \sqrt{2R}$.

Does a nonzero polynomial $Q(x, y)$ satisfying (6.4) exist, and if so, how can we find it? As in the case of the WB-decoder, this problem is reduced to linear algebra. Let us denote by $A_0, A_1, \ldots, A_\ell$ the column vector consisting of the unknown coefficients of $Q_0, Q_1, \ldots, Q_\ell$, respectively. Furthermore, let

$$V_d := \begin{pmatrix} 1 & x_1 & \cdots & x_1^{d-1} \\ 1 & x_2 & \cdots & x_2^{d-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{d-1} \end{pmatrix}, \qquad D := \begin{pmatrix} y_1 & 0 & \cdots & 0 & 0 \\ 0 & y_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & y_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & y_n \end{pmatrix}. \tag{6.5}$$

Then (6.4) is equivalent to

$$\left(D^\ell V_e \mid D^{\ell-1} V_{e+k-1} \mid \cdots \mid D V_{e+(\ell-1)(k-1)} \mid V_{e+\ell(k-1)}\right) \cdot \left(\begin{array}{c} A_\ell \\ \hline A_{\ell-1} \\ \hline \vdots \\ \hline A_1 \\ \hline A_0 \end{array}\right) = 0. \tag{6.6}$$

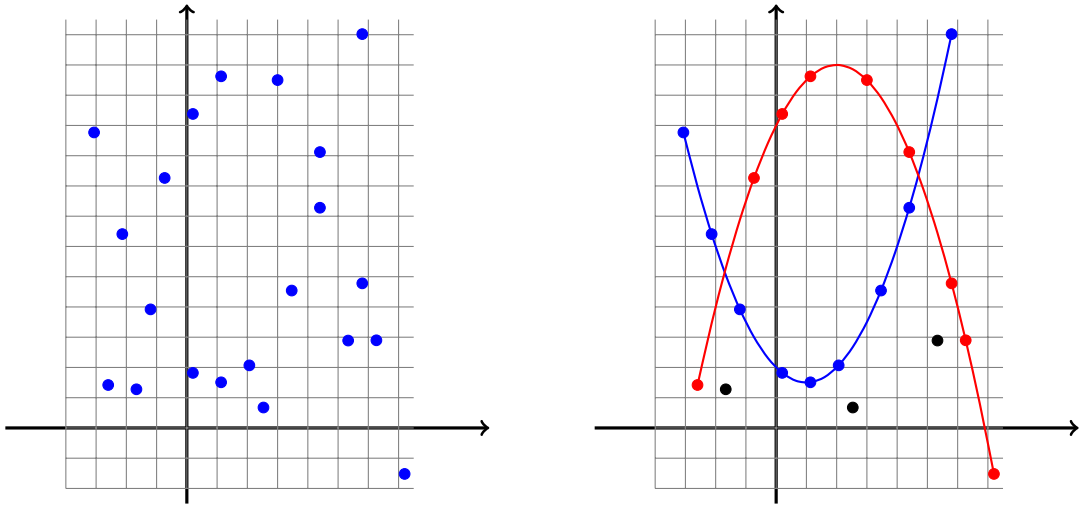
The number of unknowns in this system of equations is $(\ell+1)e + \frac{\ell(\ell+1)}{2}(k-1)$. The number of equations is n. Hence, if n is smaller than the number of unknowns, we will have a nonzero solution of this system. We therefore require

$$n \le (\ell+1)e + \frac{\ell(\ell+1)}{2}(k-1) - 1. \tag{6.7}$$

From Theorem 6.4 we know that we would be able to find all codewords of distance at most $n - e - \ell(k-1) - 1$ from the received word. The task is to maximize this value, subject to the constraint in (6.7), i.e., to minimize $e + \ell(k-1)$ subject to those constraints. The minimal value for e subject to (6.7) is $n/(\ell+1) - \ell(k-1)/2 + 1$, so that the minimal value for $e + \ell(k-1)$ is $n/(\ell+1) + \ell(k-1)/2 + 1$.

If we phrase everything relative to n, and assume that n is large, we obtain that the algorithm is capable of correcting a fraction of

$$\tau = 1 - \frac{1}{\ell+1} - \frac{\ell}{2}R$$



(a) (b)

Figure 6.3: Demonstration of Sudan's algorithm. (a) We are asked to identify all polynomials of degree at most 2 that pass through at least 9 of the 21 points shown. (b) We find two quadratic polynomials, and three spurious points which don't lie on any of the quadratic curves.

errors with a list of size at most ℓ, provided that $e/n = 1/(\ell+1) - \ell R/2$ is positive, i.e., provided that $R < \frac{2}{\ell(\ell+1)}$. For each ℓ we obtain a fraction of recoverable errors from the previous displayed equation. Figure 6.2 summarizes a plot of this function, which consists of pairwise linear functions. Also plotted is $1 - \sqrt{2R}$, which approximates the error correction capability very well for low rates. As can be seen, the new algorithm is better than the WB-algorithm if the rate is lower than $1/3$.

6.4. A Brief Note on Factoring

Sudan's list decoding algorithm and the algorithm of Guruswami-Sudan described below need to find factors of the form $y - f(x)$ of $Q(x, y)$. One possibility to do this is to factor $Q(x, y)$ completely. This can be done in polynomial time using randomized algorithms. A different approach is to pointedly look for factors of the form $y - f(x)$. Several researchers have dealt with this problem, among them Augot-Pecquet, Gao-Shokrollahi, Høholdt-Nielsen, and Roth-Rückenstein.

There are essentially two approaches: one of them is to work in a large finite field $\mathbb{F}_{q^m}$, and specialize x by a generator of that field. In other words, we take an irreducible polynomial $p(x)$ of sufficiently large degree m, and reduce $Q(x, y)$ modulo that polynomial to obtain a polynomial to obtain a univariate polynomial in y over $\mathbb{F}_{q^m}$. Next, we find roots of that univariate polynomial to obtain factors of the form $y - (f(x) \bmod p(x))$. If the degree of $p(x)$ is larger than the largest degree of $f(x)$ possible, then we can read off the right roots.

The other approach uses something very similar to Newton iteration. We specialize the variable x to some element x_0 in the field, for example 0. Next, we find the roots of $Q(0, y)$. Let the roots be $y_0, y_1, \ldots, y_t$. Then we know that if $y - f(x)$ is a factor of $Q(x, y)$, then $f(0)$ is one of these values. If $Q(0, y)$ does not have multiple roots, then, fixing $f(0)$, we can determine the higher coefficients of $f(x)$ iteratively. Details of this algorithm will be provided in the exercises.

If $Q(0, y)$ has multiple roots, then other techniques have to be used, which we are not going to discuss here.

6.5. The List Decoding Algorithm of Guruswami-Sudan

How can we correct more errors than with the algorithm of Sudan. Guruswami and Sudan had the following idea to do this: suppose that we look for a polynomial that passes through all the points (x_i, y_i) with some given multiplicity. In other words, we are looking for a polynomial of the form

$$Q(x, y) = (y - f_1(x))^r (y - f_2(x))^r \cdots (y - f_\ell(x))^r \cdot E(x)$$

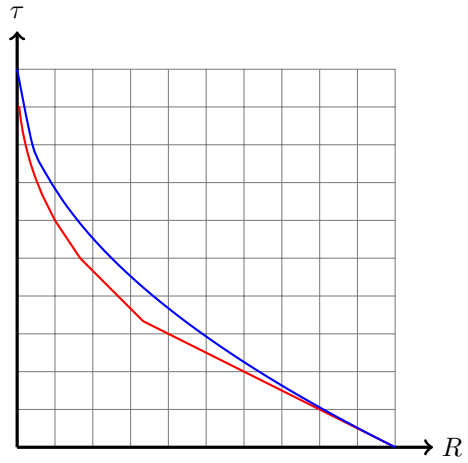


Figure 6.4: Comparison between the error correction capabilities of Sudan's algorithm (lower curve) and the algorithm of Guruswami and Sudan. The latter is better than the former and better than the WB-algorithm on the entire range for R.

describing the set of points. This polynomial has the property that for all the points (x_i, y_i), we have

$$Q(x + x_i, y + y_i) = \sum_{j \geq r} \sum_{s+k=j} q_{s,k}^{(i)} x^s y^k, \tag{6.8}$$

for some $q_{s,k}^{(i)} \in \mathbb{F}_q$. Suppose now that we have found a nonzero polynomial $Q(x, y)$ satisfying (6.8). Consider $F(x) := Q(x, f(x))$ and suppose that y and the evaluation vector of $f(x)$ agree on at least t positions. Then $F(x)$ has rt roots, since each agreement point gives rise to a zero of $F(x)$ of multiplicity r. If we can prove that we can choose $Q(x, y)$ such that the degree of $F(x)$ is less than rt, then this shows that $F(x) = 0$, and hence $(y - f(x))^r$ divides $Q(x, y)$.

To guarantee that the degree of $Q(x, f(x))$ is small, we need to design $Q(x, y)$ properly. For that, we introduce the notion of the (a, b)-degree of a monomial $x^i y^j$, defined as $ai + bj$. The normal degree is then simply the $(1, 1)$-degree. The degree of $Q(x, f(x))$ is thus at most the $(1, k - 1)$-degree of $Q(x, y)$.

Our task is to find a nonzero polynomial $Q(x, y)$ satisfying (6.8) and having a $(1, k - 1)$-degree that is as small as possible. For this, let us write

$$Q(x, y) = \sum_{i \geq 0} \sum_{j+k=i} q_{jk} x^j y^k$$

with unknown coefficients q_{ik}. Then the coefficients of $Q(x + x_i, y + y_i)$ are linear forms in q_{ik}, and Condition (6.8) translates to $r(r + 1)/2$ homogeneous equations for the q_{ik}, one for each of the "missing" monomials. In total, there are $nr(r + 1)/2$ equations.

If we assume that the $(1, k - 1)$-weighted degree of $Q(x, y)$ is less than or equal to D, then the number of monomials of which Q is potentially composed equals

$$\begin{aligned} \sum_{i=0}^{D} \sum_{j=0}^{\lfloor (D-i)/(k-1) \rfloor} 1 &= \sum_{i=0}^{D} \left\lfloor \frac{D-i}{k-1} + 1 \right\rfloor \\ &\geq \sum_{i=0}^{D} \frac{i}{k-1} \\ &= \frac{D(D+1)}{2(k-1)}. \end{aligned}$$

Thus, if we assume that

$$\frac{D(D+1)}{2(k-1)} \geq n \frac{r(r+1)}{2} + 1, \tag{6.9}$$

then we can guarantee the existence of a nonzero $Q(x, y)$ satisfying (6.8), and any polynomial $f(x)$ whose evaluation has an agreement of at least $(D + 1)/r$ with the received vector has the property that $(y - f(x))^r$ divides $Q(x, y)$.

Therefore, we would like to minimize D subject to (6.9). We loosen that condition, and require that

$$\frac{D^2}{2(k-1)} \geq n\frac{(r+1)^2}{2},$$

leading to $D \geq (r+1)\sqrt{n(k-1)}$. Choosing D to be equal to the expression on the right will lead to an agreement of at least

$$\frac{r+1}{r}(\sqrt{n(k-1)}+1).$$

Phrasing everything in terms of the error correction radius τ, this means that this algorithm is capable to find a short list of all polynomials that are within a relative distance of

$$1 - \frac{r+1}{r}\sqrt{R}$$

of the received word. Since there is no a-priori bound on r, in the limit, as r grows large, we obtain an algorithm that can "correct" up to a fraction of

$$1 - \sqrt{R}$$

errors.

Figure 6.3 shows a plot of the error correction capability of this algorithm versus the algorithm of Sudan. As can be seen, the new algorithm is better than the old one on the entire range for R. Moreover, the new algorithm is also better than the WB-algorithm on the entire range.

We finish this section with some comments on the computational complexity of this algorithm. We have to solve a system of equations with $O(nr^2)$ unknowns. Using normal Gaussian elimination, we would need $O(n^3r^6)$ operations. However, the matrix we are dealing with has structure, and using various techniques (such as the displacement method discussed later in the class), one can solve the system in $O(n^2r^4)$ operations. On the other hand, we have to choose r to be very large in order to fully benefiting from the algorithm, which may render the algorithm ineffective in practice. A Master's Thesis by Julie Marc, done at our lab, studies this algorithm from the point of view of practicability. Interested readers may want to consult that thesis (which will be available on our web site soon).

6.6. Further Developments

A natural question to ask is whether the $1 - \sqrt{R}$ bound of Guruswami-Sudan is the best possible, i.e., if there are examples in which the size of the list is too large when the number of errors is beyond this bound. The answer to this question is not known. What we do know is that extensions of Reed-Solomon codes over very large alphabets have list decoding algorithms that can decode well beyond the $1 - \sqrt{R}$ bound — almost as far as $1 - R$, which is the ultimate error correcting radius. More precisely, if the alphabet size is q^m, then one can design codes of length at most q and decoding algorithms that would be able to correct up to a fraction of $1 - \sqrt[m+1]{R^m}$ errors. For $m = 1$, we recover the Guruswami-Sudan bound, and as m goes to infinity, the error correction radius converges to $1 - R$.

The first such algorithm (with a slightly worse error correction radius than given above) was provided by to Parvaresh and Vardy, and it was vastly extended by Guruswami and Rudra to get the error correction radius reported above. These algorithms were inspired by another algorithm, due to Bleichenbacher, Kiyayias, and Yung. The latter algorithm studies Reed-Solomon for which the length is significantly smaller than the alphabet size, and provides error correction algorithms when these codes are used on the q-ary symmetric channel. These algorithms can correct almost up to a fraction of $1 - R$ of errors, with very small error probabilities. An extension of these algorithms, and a detailed analysis was provided by Brown, Minder, and the author, and is available from our web site.