2. List Decoding of Generalized Reed-Solomon Codes

Gadiel Seroussi

October 14, 2022

- The Guruswami-Sudan algorithm
- Hasse derivatives
- Guruswami-Sudan algorithm: auxiliary lemma
- Guruswami-Sudan interpolation lemma
- Guruswami-Sudan factorization lemma
- The Guruswami-Sudan (GS) decoder
- The Guruswami-Sudan algorithm: example
- Optimizing the decoding radius
- Best values of r for $\ell = 4$
- Comparison with list-1 decoder and asymptotic behavior
- The average number of incorrect codewords in the list
- Finding z-roots of bi-variate polynomials
- Algoritm BiRoot (Roth-Ruckenstein 2000)
- Algorithm BiRoot: correctness

- Algorithm BiRoot: complexity

List Decoding of Linear Codes

Given

- an $[n, k, d]$ linear code $\mathcal{C}$ over a field F,
- a channel $S = (F, F, P)$ (assuming for simplicity channel output alphabet = input alphabet),
- a positive integer ℓ.

Definition

A *list-ℓ decoder* is a mapping $\mathcal{D} : F^n \to 2^{\mathcal{C}}$ (subsets of $\mathcal{C}$), where $|\mathcal{D}(\mathbf{y})| \leq \ell$ for every $\mathbf{y} \in \mathbb{F}^n$.

Given a received word $\mathbf{y}$, the decoder returns a *list* of *at most* ℓ codewords.

List Decoding of Linear Codes

Given

- an $[n, k, d]$ linear code $\mathcal{C}$ over a field F,
- a channel $S = (F, F, P)$ (assuming for simplicity channel output alphabet = input alphabet),
- a positive integer ℓ.

Definition

A *list-ℓ decoder* is a mapping $\mathcal{D} : F^n \to 2^{\mathcal{C}}$ (subsets of $\mathcal{C}$), where $|\mathcal{D}(\mathbf{y})| \leq \ell$ for every $\mathbf{y} \in \mathbb{F}^n$.

Given a received word $\mathbf{y}$, the decoder returns a *list* of *at most ℓ* codewords.

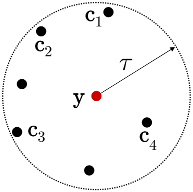
We declare success if *the list includes the transmitted codeword*.

List Decoding of Linear Codes (ii)

Definition

An integer τ is a *decoding radius* of a list-ℓ decoder $\mathcal{D}$ if

$$\mathrm{d}(\mathbf{y}, \mathbf{c}) \leq \tau \implies \mathbf{c} \in \mathcal{D}(\mathbf{y}) \quad \forall\, \mathbf{y} \in F^n,\ \mathbf{c} \in \mathcal{C}.$$



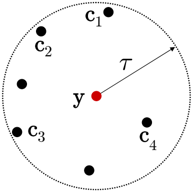
$\mathcal{D}$ captures *all* codewords at distance τ or less from any received word $\mathbf{y}$.

List Decoding of Linear Codes (ii)

Definition

An integer τ is a *decoding radius* of a list-ℓ decoder $\mathcal{D}$ if

$$\mathrm{d}(\mathbf{y}, \mathbf{c}) \leq \tau \implies \mathbf{c} \in \mathcal{D}(\mathbf{y}) \;\; \forall \, \mathbf{y} \in F^n, \; \mathbf{c} \in \mathcal{C}.$$



$\mathcal{D}$ captures *all* codewords at distance τ or less from any received word $\mathbf{y}$.

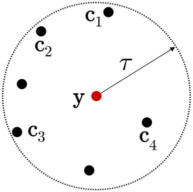
- If the number of errors that actually occurred is τ or less, then the returned list is guaranteed to contain the transmitted codeword.
- A nearest-codeword decoder for an $[n, k, d]$ linear code $\mathcal{C}$ is a list-1 decoder with decoding radius $\lfloor (d-1)/2 \rfloor$.

List Decoding of Linear Codes (ii)

Definition

An integer τ is a *decoding radius* of a list-ℓ decoder $\mathcal{D}$ if

$$\mathrm{d}(\mathbf{y}, \mathbf{c}) \le \tau \implies \mathbf{c} \in \mathcal{D}(\mathbf{y}) \;\; \forall\, \mathbf{y} \in F^n,\; \mathbf{c} \in \mathcal{C}.$$



$\mathcal{D}$ captures *all* codewords at distance τ or less from any received word $\mathbf{y}$.

- If the number of errors that actually occurred is τ or less, then the returned list is guaranteed to contain the transmitted codeword.
- A nearest-codeword decoder for an $[n, k, d]$ linear code $\mathcal{C}$ is a list-1 decoder with decoding radius $\lfloor (d-1)/2 \rfloor$.

But, what good is a list of words if what we want is the *unique* codeword that was sent?

List Decoding of Linear Codes (iii)

How do we pick the right codeword from $\mathcal{D}(\mathbf{y})$?

- Maximizing likelihood: $P(\,\mathbf{y}\ \text{ received}\,|\,\mathbf{c}\ \text{ transmitted}\,)$.
  Reduces *maximum likelihood decoding* for errors of weight up to τ to a search among ℓ codewords, with ties resolved according to some (deterministic or randomized) policy.
- Using side information on codewords, such as *a priori* probabilities (maybe derived from the context).

List Decoding of Linear Codes (iii)

How do we pick the right codeword from $\mathcal{D}(\mathbf{y})$?

- Maximizing likelihood: $P(\,\mathbf{y}\ \text{ received}\,|\,\mathbf{c}\ \text{ transmitted}\,).$
  Reduces *maximum likelihood decoding* for errors of weight up to τ to a search among ℓ codewords, with ties resolved according to some (deterministic or randomized) policy.
- Using side information on codewords, such as *a priori* probabilities (maybe derived from the context).
- *The list may contain a single codeword even when* $\tau > \lfloor (d-1)/2 \rfloor$!
  (We'll get back to this later.)

Bivariate polynomials

- $F[x,z] =$ set of all bivariate polynomials in x, z over a field F:

$$F[x,z] = \left\{ f(x,z) = \sum_{i,j=0}^{m} f_{i,j} x^i z^j \ : \ m \in \mathbb{Z}_{\geq 0},\ f_{i,j} \in F \right\}.$$

- We will also use the equivalent representation

$$F[x][z] = \left\{ f(x,z) = \sum_{j=0}^{m} f_j(x) z^j \ : \ m \in \mathbb{Z}_{\geq 0},\ f_j \in F[x] \right\}.$$

Bivariate polynomials

- $F[x, z] =$ set of all bivariate polynomials in x, z over a field F:

$$F[x, z] = \left\{ f(x, z) = \sum_{i,j=0}^{m} f_{i,j} x^i z^j \ : \ m \in \mathbb{Z}_{\geq 0},\ f_{i,j} \in F \right\}.$$

- We will also use the equivalent representation

$$F[x][z] = \left\{ f(x, z) = \sum_{j=0}^{m} f_j(x) z^j \ : \ m \in \mathbb{Z}_{\geq 0},\ f_j \in F[x] \right\}.$$

- $F[x, y]$ is a *ring*. Addition and multiplication are well defined, additive inverses exist, multiplicative ones do not in general (except for scalars).
- $F[x][z]$ is a ring of univariate polynomials with coefficients in $F[x]$, which is a ring of univariate polynomials with coefficients in F.

Bivariate polynomials: degree

Definition

The (μ, ν)-*degree* of $f(x,z) \in F[x,z]$, $\mu, \nu \geq 0$:

$$\deg_{\mu,\nu} f(x,z) = \max_{i,j \,:\, f_{i,j} \neq 0} \{i\mu + j\nu\} \, .$$

Bivariate polynomials: degree

Definition

The (μ, ν)-*degree* of $f(x,z) \in F[x,z]$, $\mu, \nu \geq 0$:

$$\deg_{\mu,\nu} f(x,z) = \max_{i,j\,:\,f_{i,j} \neq 0} \{i\mu + j\nu\} .$$

- $\deg_{1,1} f(x,z)$ is the ordinary degree of $f(x,z)$ in $F[x,z]$.
- $\deg_{0,1} f(x,z)$ is the ordinary degree of $f(x,z)$ when regarded as an element of $F[x][z]$ (degree in z).
- By convention, $\deg_{\mu,\nu}(0) = -\infty$.

Bivariate polynomials (ii)

Definition

Let $f(x,z), g(x,z) \in \mathbb{F}[x,z],\ f(x,z) \neq 0$. We say that $f(x,z)$ *divides* (or is a *factor* of) $g(x,z)$ if $g(x,z) = f(x,z)h(x,z)$ for some $h \in F[x,z]$.

- $F[x,z]$ is *not a Euclidean ring*. However, $F[x][z]$ is, and $F[x,z]$ is a *unique factorization domain*.

Bivariate polynomials (ii)

Definition

Let $f(x,z), g(x,z) \in \mathbb{F}[x,z],\ f(x,z) \neq 0$. We say that $f(x,z)$ *divides* (or is a *factor* of) $g(x,z)$ if $g(x,z) = f(x,z)h(x,z)$ for some $h \in F[x,z]$.

- $F[x,z]$ is *not a Euclidean ring*. However, $F[x][z]$ is, and $F[x,z]$ is a *unique factorization domain*.

Definitions

- A (z-)*linear factor* of $Q(x,z)$ is a factor of the form $z - f(x)$, $f(x) \in F[x]$.
- $f(x)$ is a z-*root* of $Q(x,z)$ if $Q(x, f(x)) = 0$ (identically).

Bivariate polynomials (ii)

Definition

Let $f(x,z), g(x,z) \in \mathbb{F}[x,z],\ f(x,z) \neq 0$. We say that $f(x,z)$ *divides* (or is a *factor* of) $g(x,z)$ if $g(x,z) = f(x,z)h(x,z)$ for some $h \in F[x,z]$.

- $F[x,z]$ is *not a Euclidean ring*. However, $F[x][z]$ is, and $F[x,z]$ is a *unique factorization domain*.

Definitions

- A (z-)*linear factor* of $Q(x,z)$ is a factor of the form $z - f(x)$, $f(x) \in F[x]$.
- $f(x)$ is a z-*root* of $Q(x,z)$ if $Q(x, f(x)) = 0$ (identically).

Lemma

$f(x)$ is a z-root of $Q(x,z)$ if and only if $z - f(x)$ is a factor of $Q(x,z)$.

- Proof not totally trivial since $F[x]$ is not a field—work over the *field of rational functions* $F(x)$.

GRS decoding through bivariate polynomials

- $\mathcal{C}_{\text{GRS}}$: $[n, k, d]$ GRS code over a field F. For simplicity, we assume a generator matrix of the form

$$G_{\text{GRS}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_n^{k-1} \end{pmatrix}$$

GRS decoding through bivariate polynomials

- $\mathcal{C}_{\text{GRS}}$: $[n, k, d]$ GRS code over a field F. For simplicity, we assume a generator matrix of the form

$$G_{\text{GRS}} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

- Associate $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1}) \in F^k$ with $u(x) = \sum_{i=0}^{k-1} u_i x^i \in F_k[x]$.

$$\mathcal{C}_{\text{GRS}} = \left\{ \mathbf{u}\, G_{\text{GRS}} = \big(\, u(\alpha_1),\ u(\alpha_2),\ \ldots,\ u(\alpha_n) \,\big) \ \ : \ \ u(x) \in F_k[x] \right\}$$

- Assume $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ was sent, and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ was received, with $\mathsf{d}(\mathbf{y}, \mathbf{c}) \leq \frac{1}{2}(d-1)$. Since $n \geq k$, reconstructing $\mathbf{c} = (\, u(\alpha_1), u(\alpha_2), \ldots, u(\alpha_n) \,)$ is the same as reconstructing $u(x)$.

GRS decoding through bivariate polynomials (ii)

- Construct $Q(x,z) \in F[x,z]$ satisfying *degree constraints*

$$\deg_{0,1} Q(x,z) \leq 1 \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tfrac{1}{2}(d-1)$$

and *interpolation constraints*

$$Q(\alpha_j, y_j) = 0, \quad j = 1, 2, \ldots, n \quad (*)$$

GRS decoding through bivariate polynomials (ii)

- Construct $Q(x,z) \in F[x,z]$ satisfying *degree constraints*

$$\deg_{0,1} Q(x,z) \le 1 \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tfrac{1}{2}(d-1)$$

  and *interpolation constraints*

$$Q(\alpha_j, y_j) = 0, \quad j = 1,2,\ldots,n \quad (*)$$

- The degree constraints mean that $Q(x,z)$ must be of the form

$$Q(x,z) = Q_0(x) + zQ_1(x),$$

GRS decoding through bivariate polynomials (ii)

- Construct $Q(x,z) \in F[x,z]$ satisfying *degree constraints*

$$\deg_{0,1} Q(x,z) \le 1 \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tfrac{1}{2}(d-1)$$

and *interpolation constraints*

$$Q(\alpha_j, y_j) = 0, \quad j = 1, 2, \ldots, n \quad (*)$$

- The degree constraints mean that $Q(x,z)$ must be of the form

$$Q(x,z) = Q_0(x) + zQ_1(x), \quad \boxed{k-1+\deg(Q_1) < n-\tfrac{1}{2}(d-1)}$$

$$\deg(Q_0) < n - \tfrac{1}{2}(d-1) \quad \text{and} \quad \deg Q_1(x) < \tfrac{1}{2}(d+1)$$

GRS decoding through bivariate polynomials (ii)

- Construct $Q(x,z) \in F[x,z]$ satisfying *degree constraints*

$$\deg_{0,1} Q(x,z) \le 1 \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tfrac{1}{2}(d-1)$$

  and *interpolation constraints*

$$Q(\alpha_j, y_j) = 0, \quad j = 1,2,\ldots,n \quad (*)$$

- The degree constraints mean that $Q(x,z)$ must be of the form

$$Q(x,z) = Q_0(x) + zQ_1(x), \qquad \boxed{k-1+\deg(Q_1) < n-\tfrac{1}{2}(d-1)}$$

$$\deg(Q_0) < n - \tfrac{1}{2}(d-1) \quad \text{and} \quad \deg Q_1(x) < \tfrac{1}{2}(d+1)$$

- This still allows $Q(x,z)$ to have

$$\lceil n - \tfrac{1}{2}(d-1) \rceil + \lceil \tfrac{1}{2}(d+1) \rceil \ge n+1$$

  significant (unknown) coefficients. On the other hand $(*)$ is a set of n homogeneous linear equations in these unknowns $\Longrightarrow$ *there is at least one nonzero solution $Q(x,z)$ satisfying the constraints.*

GRS decoding through bivariate polynomials (ii)

Let $n_0 = \deg(Q_0), \;\; n_1 = \deg(Q_1)$. Then

$$Q(\alpha_j, y_j) = \sum_{s=0}^{n_0} Q_{s,0}\alpha_j^s + \sum_{t=0}^{n_1} Q_{t,1}\alpha_j^t y_j \,.$$

The equations $(*)$ can be written as

$$
\begin{array}{cc}
 & \begin{array}{cc} \longleftarrow \; Q_0 \; \longrightarrow & \longleftarrow \; Q_1 \; \longrightarrow \end{array} \\
n & \left(\begin{array}{ccccc|ccccc}
1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n_0} & y_1 & \alpha_1 y_1 & \alpha_1^2 y_1 & \cdots & \alpha_1^{n_1} y_1 \\
1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n_0} & y_2 & \alpha_2 y_2 & \alpha_2^2 y_2 & \cdots & \alpha_2^{n_1} y_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{n_0} & y_n & \alpha_n y_n & \alpha_n^2 y_n & \cdots & \alpha_n^{n_1} y_n
\end{array}\right) \\
 & \longleftarrow \; n_0 + n_1 + 2 \geq n + 1 \; \longrightarrow
\end{array}
\left[\begin{array}{c}
Q_{0,0} \\ Q_{1,0} \\ Q_{2,0} \\ \vdots \\ Q_{n_0,0} \\ \hline Q_{0,1} \\ Q_{1,1} \\ Q_{2,1} \\ \vdots \\ Q_{n_1,1}
\end{array}\right] = \mathbf{0}\,.
$$

There is at least one nonzero solution $Q(x,z)$ satisfying the constraints.

GRS decoding through bivariate polynomials (iii)

There is at least one nonzero solution $Q(x,z)$ satisfying the constraints.

- Consider such a solution, and define

$$\varphi(x) = Q(x, u(x)) = Q_0(x) + u(x)Q_1(x) \quad (**)$$

- Denote the set of error locations $J = \{ j : y_j \neq c_j \}$. For $j \notin J$ we have

$$\varphi(\alpha_j) = Q(\alpha_j, u(\alpha_j)) = Q(\alpha_j, c_j) = Q(\alpha_j, y_j) \stackrel{(*)}{=} 0$$

$\Longrightarrow$ $\varphi(x)$ has at least $n - |J|$ distinct roots in F. But

$$\deg \varphi(x) \leq \max\{\deg Q_0(x), \deg u(x) + \deg Q_1(x)\} < n - \tfrac{1}{2}(d-1) \leq n - |J|$$

$\Longrightarrow$ $\varphi(x)$ must be identically zero $\Longrightarrow$ we can solve for $u(x)$ in $(**)$:

$$u(x) = -\frac{Q_0(x)}{Q_1(x)}$$

This recovers the transmitted codeword $\mathbf{c}$.

GRS decoding through bivariate polynomials (iv)

- Since $\varphi \equiv 0$, we also have, for $j \in J$

$$\begin{aligned} 0 = \varphi(\alpha_j) = Q(\alpha_j, c_j) &= Q_0(\alpha_j) + c_j Q_1(\alpha_j) \quad \text{and} \\ 0 \overset{(*)}{=} Q(\alpha, y_j) &= Q_0(\alpha_j) + y_j Q_1(\alpha_j) \\ &\Longrightarrow \underbrace{(y_j - c_j)}_{\neq 0} Q_1(\alpha_j) = 0 \end{aligned}$$

- Therefore, $Q_1(\alpha_j) = 0$ for all $j \in J$
  $\Longrightarrow Q_1(x)$ is divisible by $V(x) = \prod_{j \in J}(x - \alpha_j) = x^{|J|}\Lambda(x^{-1})$ where Λ is the *error locator polynomial* we defined for standard GRS decoding.
- In fact, $Q(x, z) = V(x)(z - u(x))$ is a solution to the degree and interpolation constraints and we have $V(x) = Q_1(x)$. This solution $Q(x, z)$ has the smallest possible $(1, k-1)$-degree and is unique up to scalar multiples.
  - The GRS decoding scheme just described is closely related to the *Welch-Berlekamp* GRS decoding algorithm.

List-ℓ decoding for $\ell > 1$: Sudan's algorithm

- Consider an $[n, k, d]$ GRS code $\mathcal{C}_{\mathrm{GRS}}$, and define

$$R' = \frac{k-1}{n}\,.$$

  It will be convenient to use R' rather than R to represent code rate.

- $\mathcal{C}_{\mathrm{GRS}}$ is MDS, so $n = k+d-1$, or $R' = 1-\delta$, where $\delta = d/n$.
- Madhu Sudan (1997) introduced a list-ℓ decoder for GRS codes, with decoding radius Δ,

$$\boxed{\Delta = \left\lceil n\,\Theta_{\ell,1}(R') \right\rceil - 1\,,}$$

  where

$$\boxed{\Theta_{\ell,1}(R') = \frac{\ell}{\ell+1} - \frac{\ell}{2}R'}$$

(The second sub-index 1 of $\Theta_{\ell,1}$ will be justified later.)

Sudan's algorithm (ii)

$$\boxed{\Theta_{\ell,1}(R') = \frac{\ell}{\ell+1} - \frac{\ell}{2}R'}$$

Sudan's algorithm (ii)

$$\boxed{\Theta_{\ell,1}(R') = \frac{\ell}{\ell+1} - \frac{\ell}{2}R'}$$

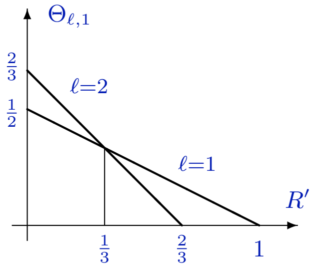
Example

$\ell = 1$: $\Theta_{1,1}(R') = (1-R')/2 = \delta/2$ corresponding to $\Delta = \lfloor (d-1)/2 \rfloor$, as expected.

Example

$\ell = 2$: $\Theta_{2,1}(R') = \frac{2}{3} - R'$, corresponding to

$$\Delta = \left\lceil \tfrac{2}{3}n \right\rceil - k = \left\lfloor \tfrac{2}{3}(n+1) \right\rfloor - k\ .$$



When $R' > \frac{1}{3}$ there is no point in selecting $\ell{=}2$ over $\ell{=}1$.

In general, choose ℓ such that

$$\Theta_{\ell,1}(R') \geq \Theta_{\ell-1,1}(R')$$
$$\Leftrightarrow\ R' \leq 2/(\ell^2 + \ell).$$

Sudan's algorithm (ii-cont.)

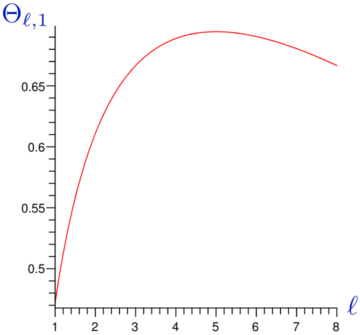
Example: GRS code with parameters $[18, 2, 17]$, $R' = 1/18$.

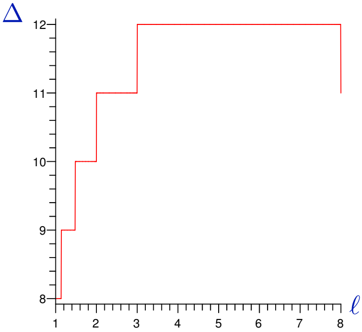
ℓ	$\Theta_{\ell,1}(R')$	Δ
1	17/36	8
2	11/18	10
3	2/3	11
4	31/45	12

Sudan's algorithm (ii-cont.)

Example: GRS code with parameters $[18, 2, 17]$, $R' = 1/18$.

ℓ	$\Theta_{\ell,1}(R')$	Δ
1	17/36	8
2	11/18	10
3	2/3	11
4	31/45	12





Sudan's algorithm (iii)

Lemma (Interpolation lemma)

Let $\ell, \tau \in \mathbb{Z}_{>0}$ *be such that* $\tau < n\Theta_{\ell,1}(R')$*. For every vector* $(y_1, y_2, \ldots, y_n) \in F^n$ *there exists a nonzero bivariate polynomial* $Q(x, z) \in F[x, z]$ *that satisfies the constraints*

$$\deg_{0,1} Q(x,z) \le \ell \quad \textit{and} \quad \deg_{1,k-1} Q(x,z) < n - \tau \ , \qquad (*)\ \text{(degree)}$$

$$Q(\alpha_j, y_j) = 0 \ , \quad j = 1, 2, \ldots, n \ . \qquad (**)\ \text{(interp.)}$$

Proof of Sudan's interpolation lemma

$$\deg_{0,1} Q(x,z) \le \ell \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tau \quad (*) \text{ (degree)}$$

Recall $\Theta_{\ell,1}(R') = \frac{\ell}{\ell+1} - \frac{\ell}{2} R'$ and $\tau < \Theta_{\ell,1}(R')$.

Proof.

$Q(x,z)$ is of degree at most ℓ in z, i.e.:

$$Q(x,z) = \sum_{t=0}^{\ell} Q_t(x) z^t .$$

Let $n_t = \deg Q_t$. Then, by the second degree constraint, we must have $t(k-1) + n_t < n - \tau$. Therefore, the number of significant coefficients allowed by $(*)$ is:

$$\sum_{t=0}^{\ell} \big((n-\tau) - t(k-1)\big) = (\ell+1)(n-\tau) - \binom{\ell+1}{2}(k-1) = (\ell+1)(n-\tau) - \binom{\ell+1}{2} nR'$$
$$= (\ell+1)\big(n - \tau - \tfrac{1}{2}\ell n R'\big)$$

Proof of Sudan's interpolation lemma

$$\deg_{0,1} Q(x,z) \le \ell \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tau \quad (*)\ (\text{degree})$$

Recall $\Theta_{\ell,1}(R') = \frac{\ell}{\ell+1} - \frac{\ell}{2} R'$ and $\tau < \Theta_{\ell,1}(R')$.

Proof.

$Q(x,z)$ is of degree at most ℓ in z, i.e.:

$$Q(x,z) = \sum_{t=0}^{\ell} Q_t(x) z^t .$$

Let $n_t = \deg Q_t$. Then, by the second degree constraint, we must have $t(k-1) + n_t < n - \tau$. Therefore, the number of significant coefficients allowed by $(*)$ is:

$$\sum_{t=0}^{\ell}((n-\tau) - t(k-1)) = (\ell+1)(n-\tau) - \binom{\ell+1}{2}(k-1) = (\ell+1)(n-\tau) - \binom{\ell+1}{2} nR'$$
$$= (\ell+1)\left(n - \tau - \tfrac{1}{2}\ell n R'\right) = (\ell+1)\left(n - \frac{n}{\ell+1} - \tau - \tfrac{1}{2}\ell n R'\right) + n$$

Proof of Sudan's interpolation lemma

$$\deg_{0,1} Q(x,z) \le \ell \quad \text{and} \quad \deg_{1,k-1} Q(x,z) < n - \tau \qquad (*) \ (\text{degree})$$

Recall $\Theta_{\ell,1}(R') = \frac{\ell}{\ell+1} - \frac{\ell}{2} R\,'$ and $\tau < \Theta_{\ell,1}(R')$.

Proof.

$Q(x,z)$ is of degree at most ℓ in z, i.e.:

$$Q(x,z) = \sum_{t=0}^{\ell} Q_t(x) z^t\,.$$

Let $n_t = \deg Q_t$. Then, by the second degree constraint, we must have $t(k-1) + n_t < n - \tau$. Therefore, the number of significant coefficients allowed by $(*)$ is:

$$\begin{aligned}\sum_{t=0}^{\ell}\big((n-\tau) - t(k-1)\big) &= (\ell+1)(n-\tau) - \tbinom{\ell+1}{2}(k-1) = (\ell+1)(n-\tau) - \tbinom{\ell+1}{2} nR' \\ &= (\ell+1)\big(n - \tau - \tfrac{1}{2}\ell nR'\big) = (\ell+1)\Big(n - \frac{n}{\ell+1} - \tau - \tfrac{1}{2}\ell nR'\Big) + n \\ &= (\ell+1)\Big(\tfrac{\ell}{\ell+1} n - \tfrac{1}{2}\ell nR' - \tau\Big) + n = (\ell+1)\Big(n\,\Theta_{\ell,1}(R') - \tau\Big) + n > n.\end{aligned}$$

Hence, there must be at least one nontrivial solution to $(**)$.

Sudan's algorithm (iv)

Lemma (Factorization lemma)

Let $Q(x,z) \in F[x,z] \setminus \{0\}$ *satisfy* $(*)$–$(**)$ *for some* τ *and* $\mathbf{y}$. *Suppose there exists* $u(x) \in F_k[x]$ *such that* $\mathbf{c} = (u(\alpha_i))_{i=1}^n$ *satisfies* $\mathrm{d}(\mathbf{y},\mathbf{c}) \leq \tau$. *Then* $(z - u(x)) \mid Q(x,z)$.

Sudan's algorithm (iv)

Lemma (Factorization lemma)

Let $Q(x,z) \in F[x,z] \setminus \{0\}$ satisfy $(*)$–$(**)$ for some τ and $\mathbf{y}$. Suppose there exists $u(x) \in F_k[x]$ such that $\mathbf{c} = (u(\alpha_i))_{i=1}^n$ satisfies $\mathsf{d}(\mathbf{y},\mathbf{c}) \le \tau$. Then $(z - u(x)) \,|\, Q(x,z)$.

Proof.

Let $J = \{\, j \,:\, y_j \neq u(\alpha_j) \,\}$ and define $\varphi(x) = Q(x, u(x))$. We have

$$\deg \varphi(x) \;\le\; \deg_{1,k-1} Q(x,z) \;\stackrel{(*)}{<}\; n - \tau \;\stackrel{\mathsf{d}(\mathbf{y},\mathbf{c}) \le \tau}{\le}\; n - |J| .$$

On the other hand, for all location indices $j \notin J$,

$$\varphi(\alpha_j) = Q(\alpha_j, u(\alpha_j)) \;\stackrel{(**)}{=}\; Q(\alpha_j, y_j) = 0 \,.$$

As before, we conclude that $\varphi(x) \equiv 0$, and, thus, $u(x)$ is a z-root of $Q(x,z)$.

A list-ℓ decoder for $\mathcal{C}_{\text{GRS}}$

A list-ℓ decoder for $\mathcal{C}_{\text{GRS}}$ derives immediately from the interpolation and factorization lemmas above.

Input: received word $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, $\ell =$ list size.
(Assume decoding radius $\tau = \lceil n\Theta_{\ell,1}(R') \rceil - 1$.)

Output: list of up to ℓ codewords $\mathbf{c} \in \mathcal{C}_{\text{GRS}}$.

1. *Interpolation step:* find a nonzero $Q(x, z) \in F[x, z]$ satisfying

$$\deg_{0,1} Q(x,z) \leq \ell \,, \quad \deg_{1,k-1} Q(x,z) \leq n\bigl(1 - \Theta_{\ell,1}(R')\bigr),$$

$$\text{and} \quad Q(\alpha_j, y_j) = 0, \quad j = 1, 2, \ldots, n.$$

2. *Factorization step:* Compute the set U of all polynomials $u(x) \in F_{nR'+1}[x]$ such that $(z - u(x)) \,|Q(x, z)$.
3. Output all the codewords $\mathbf{c} = (u(\alpha_1), u(\alpha_2), \ldots, u(\alpha_n))$ corresponding to $u(x) \in U$ such that $\mathsf{d}(\mathbf{y}, \mathbf{c}) < n\Theta_{\ell,1}(R')$.

Reverse engineering

- $\deg_{0,1} Q(x,z) \leq \ell$ and $\deg_{1,k-1} Q(x,z) < n - \tau$

  Degree constraints, through interpolation/factorization lemmas, ensure we can catch up to ℓ codewords at distance τ or less from received word.

Reverse engineering

- $\deg_{0,1} Q(x,z) \le \ell$ and $\deg_{1,k-1} Q(x,z) < n - \tau$

  Degree constraints, through interpolation/factorization lemmas, ensure we can catch up to ℓ codewords at distance τ or less from received word.
- Count free coefficients allowed by degree constraints.

$$Q(x,z) = \sum_{i=0}^{\ell} Q_i(x) z^i, \qquad \deg Q_i < n - \tau - (k-1)i$$

$$\begin{aligned} N_{\text{coeffs}} &= \sum_{i=0}^{\ell} (\, n - \tau - (k-1)i \,) \\ &= (\ell+1)(n-\tau) - (k-1)\frac{\ell(\ell+1)}{2} > n \quad \text{(we want)} \end{aligned}$$

Reverse engineering

- $\deg_{0,1} Q(x,z) \le \ell$ and $\deg_{1,k-1} Q(x,z) < n - \tau$

  Degree constraints, through interpolation/factorization lemmas, ensure we can catch up to ℓ codewords at distance τ or less from received word.
- Count free coefficients allowed by degree constraints.

$$Q(x,z) = \sum_{i=0}^{\ell} Q_i(x) z^i, \qquad \deg Q_i < n - \tau - (k-1)i$$

$$\begin{aligned} N_{\text{coeffs}} &= \sum_{i=0}^{\ell} (\, n - \tau - (k-1)i\,) \\ &= (\ell+1)(n-\tau) - (k-1)\frac{\ell(\ell+1)}{2} > n \quad \text{(we want)} \\ &\iff \quad \frac{\tau}{n} < \underbrace{\frac{\ell}{\ell+1} - \frac{\ell}{2} R'}_{\Theta_{\ell,1}(R')} \end{aligned}$$

Analysis of the computation

List-ℓ decoding can be done in polynomial time.

Analysis of the computation

List-ℓ decoding can be done in polynomial time.

1. *Interpolation step:* find a nonzero $Q(x,z) \in F[x,z]$ satisfying
$$\deg_{0,1} Q(x,z) \le \ell \,, \quad \deg_{1,k-1} Q(x,z) \le n\big(1 - \Theta_{\ell,1}(R')\big),$$
and $\quad Q(\alpha_j, y_j) = 0, \quad j = 1, 2, \ldots, n.$

Gaussian elimination: $O(n^3)$ operations in F.

Analysis of the computation

List-ℓ decoding can be done in polynomial time.

1. *Interpolation step:* find a nonzero $Q(x, z) \in F[x, z]$ satisfying
$$\deg_{0,1} Q(x, z) \leq \ell \ , \quad \deg_{1,k-1} Q(x, z) \leq n\big(1 - \Theta_{\ell,1}(R')\big),$$
$$\text{and} \quad Q(\alpha_j, y_j) = 0, \quad j = 1, 2, \ldots, n.$$

Gaussian elimination: $O(n^3)$ operations in F.

2. *Factorization step:* Compute the set U of all polynomials $u(x) \in F_{nR'+1}[x]$ such that $(z - u(x)) \,|Q(x, z)$.

Nontrivial, because the roots sought are in $F(x)$. Efficient solutions exist [Gao-Shokrollahi 1999, Roth-Ruckenstein 2000].

Analysis of the computation

List-ℓ decoding can be done in polynomial time.

1. *Interpolation step:* find a nonzero $Q(x,z) \in F[x,z]$ satisfying
$$\deg_{0,1} Q(x,z) \le \ell\ , \quad \deg_{1,k-1} Q(x,z) \le n\big(1-\Theta_{\ell,1}(R')\big),$$
$$\text{and} \quad Q(\alpha_j, y_j) = 0, \quad j = 1, 2, \ldots, n.$$

Gaussian elimination: $O(n^3)$ operations in F.

2. *Factorization step:* Compute the set U of all polynomials $u(x) \in F_{nR'+1}[x]$ such that $(z - u(x))\,|Q(x,z)$.

Nontrivial, because the roots sought are in $F(x)$. Efficient solutions exist [Gao-Shokrollahi 1999, Roth-Ruckenstein 2000].

3. Output all the codewords $\mathbf{c} = (u(\alpha_1), u(\alpha_2), \ldots, u(\alpha_n))$ corresponding to $u(x) \in U$ such that $\mathsf{d}(\mathbf{y},\mathbf{c}) < n\Theta_{\ell,1}(R')$.

Getting $\mathbf{c}$ from $u(x)$ takes $O(nk)$ operations.
Doing it for all codewords in the list takes $O(\ell nk)$ operations.

Sudan's algorithm: small example

List-2 decoder for GRS $[7,2,6]$ over $F = \mathrm{GF}(7)$ $\quad (\ell{=}2,\ R'{=}1/7,\ \lfloor \frac{d-1}{2} \rfloor{=}2)$

- Code locators $\alpha_j = j,\ j = 0, 1, \ldots, 6$.

$$G_{\mathrm{GRS}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}.$$

- Decoding radius:

$$\tau = \lceil n\Theta_{\ell,1} \rceil - 1 = \left\lceil n \left(\frac{\ell}{\ell+1} - \frac{\ell}{2} R' \right) \right\rceil - 1 = \left\lceil 7 \left(\frac{2}{3} - \frac{1}{7} \right) \right\rceil - 1 = 3\,.$$

- Degree constraints on Q: $\deg_{0,1} Q \le 2,\ \ \deg_{1,1} Q < n - \tau = 4$.

$$Q(x,z) = (q_{00} + q_{10}x + q_{20}x^2 + q_{30}x^3) + (q_{01} + q_{11}x + q_{21}x^2)z + (q_{02} + q_{12}x)z^2$$

9 variables

- Sent word: $[0000000]$ Received: $[1110000]$
- Interpolation constraints: $Q(\alpha_j, y_j) = 0, \quad 1 \le j \le 7$.

$$[1, \alpha_j, \alpha_j^2, \alpha_j^3, y_j, \alpha_j y_j, \alpha_j^2 y_j, y_j^2, \alpha_j y_j^2] \cdot [q_{00}, q_{10}, q_{20}, q_{30}, q_{01}, q_{11}, q_{21}, q_{02}, q_{12}]' \;=\; 0$$

Sudan's algorithm: small example

$$[1, \alpha_j, \alpha_j^2, \alpha_j^3, y_j, \alpha_j y_j, \alpha_j^2 y_j, y_j^2, \alpha_j y_j^2] \ \cdot \mathbf{q}' = 0$$

$$\mathbf{q} = [q_{00}, q_{10}, q_{20}, q_{30}, q_{01}, q_{11}, q_{21}, q_{02}, q_{12}]$$

$$\begin{array}{cc} & \longleftarrow\ 9\ \longrightarrow \\ 7 & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 4 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 1 & 6 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \cdot \mathbf{q}' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Solutions: $r[0,0,0,0,6,0,0,1,0] + s[0,0,0,0,0,6,0,0,1], \quad r, s \in F$.
- Set $r = 1,\ s = 0$: $\quad \mathbf{q} = [0,0,0,0,6,0,0,1,0]$
- $Q(x,z) = 6z + z^2 = z^2 - z = z(z-1)$, roots $u(x) = 0,\ u(x) = 1$.
- $u(x) = 1$ corresponds to codeword $[1,1,1,1,1,1,1]$, at distance $4 > \tau$ from $\mathbf{y}$: *discarded*.
- Codeword list: $\{\, [0,0,0,0,0,0,0] \,\}$

Sudan's algorithm: small example

$$[1, \alpha_j, \alpha_j^2, \alpha_j^3, y_j, \alpha_j y_j, \alpha_j^2 y_j, y_j^2, \alpha_j y_j^2] \; \cdot \mathbf{q}' = 0$$

$$\mathbf{q} = [q_{00}, q_{10}, q_{20}, q_{30}, q_{01}, q_{11}, q_{21}, q_{02}, q_{12}]$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 4 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 1 & 6 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \mathbf{q}' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Solutions: $r[0,0,0,0,6,0,0,1,0] + s[0,0,0,0,0,6,0,0,1], \;\; r, s \in F$.

  $Q(x,z) = 6rz + 6sxz + rz^2 + sxz^2 = (r+sx)(z^2 - z) = (r+sx)z(z-1)$,

  roots $u(x) = 0, \; u(x) = 1$.
- Codeword list: $\{\, [0,0,0,0,0,0,0] \,\}$

Sudan's algorithm: bigger example

List-4 decoder for GRS $[18, 2, 17]$ over $F = \mathrm{GF}(19)$ $\quad (\ell{=}4,\ R'{=}\frac{1}{18},\ \lfloor\frac{d-1}{2}\rfloor{=}8)$

- Code locators $\alpha_j = j,\ j = 1, 2, \ldots, 18$.

$$G_{\mathrm{GRS}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \end{pmatrix}.$$

- Decoding radius:

$$\tau = \lceil n\Theta_{\ell,1} \rceil - 1 = \left\lceil n \left(\frac{\ell}{\ell+1} - \frac{\ell}{2} R' \right) \right\rceil - 1 = \left\lceil 18 \left(\frac{4}{5} - \frac{1}{9} \right) \right\rceil - 1 = 12\,.$$

- Degree constraints on Q: $\deg_{0,1} Q \le 4,\ \ \deg_{1,1} Q < n - \tau = 6$.

$$Q(x,z) = \sum_{i=0}^{4} \left(\sum_{j=0}^{5-i} f_{i,j} x^j \right) z^i \qquad \text{20 indeterminates}\,.$$

- Assume the transmitted codeword $\mathbf{c}$ corresponds to $u(x) = 18 + 14x$, i.e.,

$$\mathbf{c} = (\,13,\ 8,\ 3, 17, 12,\ 7,\ 2, 16, 11, 6,\ 1, 15, 10,\ 5,\ 0, 14,\ 9,\ 4\,)\,,$$

and the received word is

$$\mathbf{y} = (\ 5,\ 5,\ 1, 10, 10,\ 7,\ 2, 18,\ 6,\ 6,\ 1, 15, 13,\ 5, 14,\ 3,\ 1,\ 0\,)\,.$$

The gory details

The Guruswami-Sudan algorithm

- The decoding radius of Sudan's algorithm can be increased by considering also the *derivatives* of $Q(x, z)$
- The quantity $\Theta_{\ell,1}(R')$ will be generalized to

$$\boxed{\Theta_{\ell,r}(R') = \frac{1}{(\ell+1)r}\left(\binom{\ell+1}{2}(1-R') - \binom{\ell+1-r}{2}\right)}$$

or, equivalently,

$$\Theta_{\ell,r}(R') = 1 - \frac{r+1}{2(\ell+1)} - \frac{\ell}{2r}\,R', \quad r \leq \ell\,.$$

- As before, $R' \mapsto \Theta_{\ell,r}(R')$ represents a line in the real plane. When $r = 1$, the expression reduces to the previous definition of $\Theta_{\ell,1}(R')$.
- The additional parameter r will be optimized to obtain the largest possible decoding radius.

Hasse derivatives

- We saw finite field derivatives in the computation of error values in GRS decoding, e.g.: $e_j = -\frac{\alpha_j}{v_j} \cdot \frac{\Gamma(\alpha_j^{-1})}{\Lambda'(\alpha_j^{-1})}$
  - Finite field derivatives have some familiar properties, e.g., β is a multiple root of $f(x)$ iff $f(\beta) = f'(\beta) = 0$.
  - But, in characteristic p, $f^{(p)}(x) \equiv 0$ for all f. E.g., $f''(x) \equiv 0$ in characteristic 2. Not good for characterizing root multiplicity.

Hasse derivatives

- We saw finite field derivatives in the computation of error values in GRS decoding, e.g.: $e_j = -\frac{\alpha_j}{v_j} \cdot \frac{\Gamma(\alpha_j^{-1})}{\Lambda'(\alpha_j^{-1})}$
  - Finite field derivatives have some familiar properties, e.g., β is a multiple root of $f(x)$ iff $f(\beta) = f'(\beta) = 0$.
  - But, in characteristic p, $f^{(p)}(x) \equiv 0$ for all f. E.g., $f''(x) \equiv 0$ in characteristic 2. Not good for characterizing root multiplicity.

Definition (Hasse derivative)

Let $a(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial in $F[x]$. The *ℓth Hasse derivative* of $a(x)$, denoted $a^{[\ell]}(x)$, is defined as

$$a^{[\ell]}(x) = \sum_{i=\ell}^{n} \binom{i}{\ell} a_i x^{i-\ell} \, .$$

$\binom{i}{\ell} \triangleq 0$ when $i < \ell$

Hasse derivatives

- We saw finite field derivatives in the computation of error values in GRS decoding, e.g.: $e_j = -\frac{\alpha_j}{v_j} \cdot \frac{\Gamma(\alpha_j^{-1})}{\Lambda'(\alpha_j^{-1})}$
  - Finite field derivatives have some familiar properties, e.g., β is a multiple root of $f(x)$ iff $f(\beta) = f'(\beta) = 0$.
  - But, in characteristic p, $f^{(p)}(x) \equiv 0$ for all f. E.g., $f''(x) \equiv 0$ in characteristic 2. Not good for characterizing root multiplicity.

Definition (Hasse derivative)

Let $a(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial in $F[x]$. The *ℓth Hasse derivative* of $a(x)$, denoted $a^{[\ell]}(x)$, is defined as

$$a^{[\ell]}(x) = \sum_{i=\ell}^{n} \binom{i}{\ell} a_i x^{i-\ell} .$$

$\binom{i}{\ell} \triangleq 0$ when $i < \ell$

Example: $f(x) = x^4 + x^3 + 1 \in \mathrm{GF}(2)[x]$.

$f^{[1]}(x) = \binom{4}{1}x^3 + \binom{3}{1}x^2 = x^2 = f^{(1)}(x)$, $\quad f^{[2]}(x) = \binom{4}{2}x^2 + \binom{3}{2}x = x$,

$f^{[3]}(x) = \binom{4}{3}x + \binom{3}{3} = 1$, $\quad f^{[4]}(x) = 1$

$f^{[5]}(x) = 0$

Hasse derivatives (ii)

- Properties
  - $a^{[1]}(x) = a^{(1)}(x)\,.$
  - $\left(a(x)+b(x)\right)^{[\ell]} = a^{[\ell]}(x)+b^{[\ell]}(x), \quad \left(c\cdot a(x)\right)^{[\ell]} = c\cdot a^{[\ell]}(x)$ linear.
  - $\left(a(x)b(x)\right)^{[\ell]} = \sum_{m=0}^{\ell} a^{[m]}(x) b^{[\ell-m]}(x)\,.$

Hasse derivatives (ii)

- Properties
  - $a^{[1]}(x) = a^{(1)}(x)$.
  - $(a(x)+b(x))^{[\ell]} = a^{[\ell]}(x)+b^{[\ell]}(x), \quad (c{\cdot}a(x))^{[\ell]} = c{\cdot}a^{[\ell]}(x)$ linear.
  - $(a(x)b(x))^{[\ell]} = \sum_{m=0}^{\ell} a^{[m]}(x)b^{[\ell-m]}(x)$.

Proposition

Let β be a root of $f(x) \in F[x]$ in some extension of F. The multiplicity of β as a root of f is exactly m iff

$$f^{[\ell]}(x)\Big|_{x=\beta} = 0, \ \ell = 0, 1, \ldots, m-1, \text{ and } f^{[m]}(x)\Big|_{x=\beta} \neq 0.$$

- **Example:**
  $f(x) = x^4 + 1 = (x+1)^4 \in GF(2)[x]$ vanishes at $x = 1$
  $f^{[1]}(x) = 0$ vanishes at $x = 1$
  $f^{[2]}(x) = 0$ vanishes at $x = 1$
  $f^{[3]}(x) = 0$ vanishes at $x = 1$
  $f^{[4]}(x) = 1$ does not vanish at $x = 1$

Hasse derivatives for bivariate polynomials

Definition (Hasse derivative for bivariate polynomials)

The (s,t)th Hasse derivative of $a(x,z) \in F[x,z]$ is defined as

$$a^{[s,t]}(x,z) = \sum_{i,j} \binom{i}{s}\binom{j}{t} a_{i,j} x^{i-s} z^{j-t} \ .$$

$\binom{h}{m} \triangleq 0$ when $h < m$

Guruswami-Sudan algorithm: auxiliary lemma

- Define $\mathrm{T}(r) = \{\, (s,t) \;:\; s,t \in \mathbb{Z}_{\geq 0},\; s+t < r \,\}$ notice: $|\mathrm{T}(r)| = \binom{r+1}{2}$.

Guruswami-Sudan algorithm: auxiliary lemma

- Define $\mathrm{T}(r) = \{\, (s,t) \;:\; s,t \in \mathbb{Z}_{\geq 0},\; s+t < r \,\}$ notice: $|\mathrm{T}(r)| = \binom{r+1}{2}$.

Lemma (auxiliary)

Given $u(x) \in F[x]$ and $a(x,z) \in F[x,z]$, let β and γ be elements of F such that $u(\beta) = \gamma$ and

$$a^{[s,t]}(x,z)|_{(x,z)=(\beta,\gamma)} = 0 \quad \textit{for all } (s,t) \in \mathrm{T}(r) .$$

Then $(x-\beta)^r \mid a(x,u(x))$.

Guruswami-Sudan algorithm: auxiliary lemma

- Define $\mathrm{T}(r) = \{\, (s,t) \;:\; s,t \in \mathbb{Z}_{\geq 0},\; s+t < r \,\}$ notice: $|\mathrm{T}(r)| = \binom{r+1}{2}$.

Lemma (auxiliary)

Given $u(x) \in F[x]$ and $a(x,z) \in F[x,z]$, let β and γ be elements of F such that $u(\beta) = \gamma$ and

$$a^{[s,t]}(x,z)|_{(x,z)=(\beta,\gamma)} = 0 \quad \textit{for all } (s,t) \in \mathrm{T}(r) \ .$$

Then $(x-\beta)^r \mid a(x,u(x))$.

Proof.

Define $b(v,w) = a(v{+}\beta, w{+}\gamma) \triangleq \sum_{s,t} b_{s,t} v^s w^t$. We have

$$a(v{+}\beta, w{+}\gamma) = \sum_{i,j} a_{ij} (v{+}\beta)^i (w{+}\gamma)^j = \sum_{ij} a_{ij} \sum_{s=0}^{i} \sum_{t=0}^{j} \binom{i}{s}\binom{j}{t} \beta^{i-s} \gamma^{j-t} v^s w^t$$

Equating coefficients, we get

$$b_{s,t} = \sum_{i,j} \binom{i}{s}\binom{j}{t} a_{i,j} \beta^{i-s} \gamma^{j-t} = a^{[s,t]}(x,z)|_{x=\beta, z=\gamma}$$

and, so, $b_{s,t} = 0$ for every $(s,t) \in \mathrm{T}(r)$. Hence,

$$a(x,u(x)) = b(x-\beta, u(x)-\gamma) = \sum_{s,t \,:\, s+t \geq r} b_{s,t} (x-\beta)^s (u(x)-\gamma)^t \ .$$

The result follows by observing that $(x-\beta) \mid (u(x)-\gamma)$.

Guruswami-Sudan interpolation lemma

Lemma (Guruswami-Sudan interpolation lemma)

Let ℓ, r, n, $k = nR' + 1$, *and* τ *be positive integers such that* $r \le \ell$, $k \le n$, *and* $\tau < n\,\Theta_{\ell,r}(R')$. *For every vector* $\mathbf{y} \in F^n$ *there exists a nonzero* $Q(x,z) \in F[x,z]$ *satisfying*

$$\deg_{0,1} Q(x,z) \le \ell\ , \quad \deg_{1,k-1} Q(x,z) < r(n-\tau)\ , \qquad (\star)$$

and

$$Q^{[s,t]}(x,z)|_{(x,z)=(\alpha_j,y_j)} = 0\ , \quad j = 1,2,\ldots,n\ ,\ (s,t)\in\mathrm{T}(r)\ . \qquad (\star\star)$$

Guruswami-Sudan interpolation lemma (proof)

$$\deg_{0,1} Q(x,z) \le \ell \,, \quad \deg_{1,k-1} Q(x,z) < r(n-\tau) \,, \qquad (\bigstar)$$

$$\Theta_{\ell,r}(R') = \frac{1}{(\ell+1)r}\left(\tbinom{\ell+1}{2}(1-R') - \tbinom{\ell+1-r}{2}\right)$$

Proof.

Similar to the proof for Sudan's algorithm. The number of free coefficients allowed by the degree constraints is

$$\begin{aligned}\textstyle\sum_{t=0}^{\ell}\big(r(n-\tau)-t(k-1)\big) &= (\ell+1)r(n-\tau) - \tbinom{\ell+1}{2}(k-1)\\ &= (\ell+1)r(n-\tau) - \tbinom{\ell+1}{2}nR'\\ &= \tbinom{\ell+1}{2}n(1-R') + \left((\ell+1)r - \tbinom{\ell+1}{2}\right)n - (\ell+1)r\tau\end{aligned}$$

Guruswami-Sudan interpolation lemma (proof)

$$\deg_{0,1} Q(x,z) \le \ell \,, \quad \deg_{1,k-1} Q(x,z) < r(n-\tau)\,, \qquad (\bigstar)$$

$$\Theta_{\ell,r}(R') = \frac{1}{(\ell+1)r}\left(\binom{\ell+1}{2}(1-R') - \binom{\ell+1-r}{2}\right)$$

Proof.

Similar to the proof for Sudan's algorithm. The number of free coefficients allowed by the degree constraints is

$$\begin{aligned}
\textstyle\sum_{t=0}^{\ell}\big(r(n-\tau) - t(k-1)\big) &= (\ell+1)r(n-\tau) - \binom{\ell+1}{2}(k-1) \\
&= (\ell+1)r(n-\tau) - \binom{\ell+1}{2}nR' \\
&= \binom{\ell+1}{2}n(1-R') + \left((\ell+1)r - \binom{\ell+1}{2}\right)n + (\ell+1)r\tau \\
&= \binom{\ell+1}{2}n(1-R') - \left(\binom{\ell+1-r}{2} - \binom{r+1}{2}\right)n - (\ell+1)r\tau
\end{aligned}$$

Guruswami-Sudan interpolation lemma (proof)

$$\deg_{0,1} Q(x,z) \le \ell\ , \quad \deg_{1,k-1} Q(x,z) < r(n-\tau)\ , \qquad (\bigstar)$$

$$\Theta_{\ell,r}(R') = \frac{1}{(\ell+1)r}\left(\binom{\ell+1}{2}(1-R') - \binom{\ell+1-r}{2}\right)$$

Proof.

Similar to the proof for Sudan's algorithm. The number of free coefficients allowed by the degree constraints is

$$\begin{aligned}
&\textstyle\sum_{t=0}^{\ell}\big(r(n-\tau) - t(k-1)\big) = (\ell+1)r(n-\tau) - \binom{\ell+1}{2}(k-1) \\
&= \quad (\ell+1)r(n-\tau) - \binom{\ell+1}{2}nR' \\
&= \quad \binom{\ell+1}{2}n(1-R') + \left((\ell+1)r - \binom{\ell+1}{2}\right)n - (\ell+1)r\tau \\
&= \quad \binom{\ell+1}{2}n(1-R') - \left(\binom{\ell+1-r}{2} - \binom{r+1}{2}\right)n - (\ell+1)r\tau \\
&= \quad (\ell+1)r\Big(n\,\Theta_{\ell,r}(R') - \tau\Big) + \binom{r+1}{2}n > \binom{r+1}{2}n = |\mathrm{T}(r)|n\ .
\end{aligned}$$

Thus, the interpolation constraints have at least one nontrivial solution.

Guruswami-Sudan factorization lemma

Lemma (Guruswami-Sudan factorization lemma)

Let a nonzero $Q(x, z) \in F[x, z]$ *satisfy the degree and interpolation constraints of the previous lemma for* $r, \tau \in \mathbb{Z}_{>0}$, *and a word* $\mathbf{y} \in F^n$. *Suppose there exists* $u(x) \in F_k[x]$ *such that the respective codeword,* $\mathbf{c} = \big(u(\alpha_i)\big)_{i=1}^{n}$ *satisfies* $\mathsf{d}(\mathbf{y}, \mathbf{c}) \le \tau$. *Then* $(z - u(x)) \,|\, Q(x, z)$.

Guruswami-Sudan factorization lemma

Lemma (Guruswami-Sudan factorization lemma)

Let a nonzero $Q(x,z) \in F[x,z]$ satisfy the degree and interpolation constraints of the previous lemma for $r, \tau \in \mathbb{Z}_{>0}$, and a word $\mathbf{y} \in F^n$. Suppose there exists $u(x) \in F_k[x]$ such that the respective codeword, $\mathbf{c} = \big(u(\alpha_i)\big)_{i=1}^{n}$ satisfies $\mathsf{d}(\mathbf{y},\mathbf{c}) \le \tau$. Then $(z-u(x)) \,|\, Q(x,z)$.

Proof.

Let $\overline{J}$ be the set of indexes j for which $u(\alpha_j) = y_j$. By $(\star\star)$ and the auxiliary lemma we obtain

$$(x-\alpha_j)^r \,|\, Q(x,u(x)) \,, \quad j \in \overline{J}$$

$$\Rightarrow \quad \Big(\prod_{j \in \overline{J}} (x-\alpha_j)^r\Big) \,|\, Q(x,u(x)) \,. \qquad (\star\star\star)$$

On the other hand, by $(\star)$

$$\deg Q(x,u(x)) \le \deg_{1,k-1} Q(x,z) < r(n-\tau) \le r|\overline{J}| \,.$$

Combining this with $(\star\star\star)$ we conclude that $Q(x,u(x))$ is identically zero. The result now follows from the lemma on z-roots.

The Guruswami-Sudan (GS) decoder

Input: received word $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, $\ell =$ list size.
(Assume decoding radius $\tau = \lceil n\Theta_{\ell,r}(R') \rceil - 1$.)

Output: list of up to ℓ codewords $\mathbf{c} \in \mathcal{C}_{\mathrm{GRS}}$.

1. *Interpolation step:* find a nonzero $Q(x,z) \in F[x,z]$ satisfying

$$\deg_{0,1} Q(x,z) \leq \ell\ , \quad \deg_{1,k-1} Q(x,z) \leq n\big(1 - \Theta_{\ell,r}(R')\big),$$

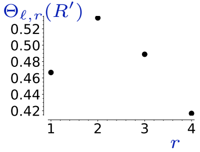
and $$Q^{[s,t]}(x,z)|_{(x,z)=(\alpha_j,y_j)} = 0\ , \quad j = 1,2,\ldots,n\ ,\ (s,t) \in \mathrm{T}(r)$$

2. *Factorization step:* Compute the set U of all polynomials $u(x) \in F_{nR'+1}[x]$ such that $z - u(x) | Q(x,z)$.
3. Output all the codewords $\mathbf{c} = (u(\alpha_1), u(\alpha_2), \ldots, u(\alpha_n))$ corresponding to $u(x) \in U$ such that $\mathsf{d}(\mathbf{y}, \mathbf{c}) < n\Theta_{\ell,r}(R')$.

The algorithm is parametrized in $r \leq \ell$. What is the best value?

The Guruswami-Sudan algorithm: example

List-4 decoder for GRS $[18, 4, 15]$ over $F = \mathrm{GF}(19)$



- Parameters: $R' = (k-1)/n = 1/6$, $\ell = 4$. The function
$$\Theta_{\ell,r}(R') = 1 - \frac{r+1}{2(\ell+1)} - \frac{\ell}{2r} R', \quad r \leq \ell$$
is maximized at $r = 2$, yielding $\Theta = 8/15$ and a decoding radius
$$\tau = \lceil n\Theta(\ell, r) \rceil - 1 = 9$$
(compare with $(d-1)/2 = 7$).
- Degree constraints on Q: $\deg_{0,1} Q \leq 4$, $\deg_{1,3} Q < r(n-\tau) = 18$.
$$Q(x,z) = \sum_{j=0}^{4} \left(\sum_{i=0}^{17-3*j} f_{i,j} x^i \right) z^j \qquad \text{60 indeterminates}\,.$$
- Assume the transmitted codeword $\mathbf{c}$ corresponds to $u(x) = 18 + 14x + 3x^2 + x^3$, i.e.,
$$\mathbf{c} = (\,17,\,9,\,0,\,15,\,3,\,8,\,17,\,17,\,14,\,14,\,4,\,9,\,16,\,12,\,3,\,14,\,13,\,6\,)\,.$$
error vector
$$\mathbf{e} = (15, 9, 0, 0, 9, 17, 0, 8, 4, 0, 0, 0, 0, 4, 0, 7, 0, 12) \quad \text{(weight 9)}$$

The gory details

Optimizing the decoding radius

- We can optimize over r to obtain the best possible decoding radius for the GS decoder. Define

$$\Theta_\ell(R') = \max_{1 \le r \le \ell} \Theta_{\ell,r}(R') \,.$$

- Define $\Upsilon_{\ell,r} = \frac{r(r-1)}{\ell(\ell+1)}$. It can be shown that

$$\Theta_{\ell,r}(R') \ge \Theta_{\ell,r-1}(R') \quad \Longleftrightarrow \quad R' \ge \Upsilon_{\ell,r} \,.$$

$\Longrightarrow$ Optimal r is the unique integer satisfying
$\Upsilon_{\ell,r} \le R' < \Upsilon_{\ell,r+1}$ (best r is a function of R' and ℓ).

Optimizing the decoding radius

- We can optimize over r to obtain the best possible decoding radius for the GS decoder. Define

$$\Theta_\ell(R') = \max_{1 \le r \le \ell} \Theta_{\ell,r}(R') \,.$$

- Define $\Upsilon_{\ell,r} = \frac{r(r-1)}{\ell(\ell+1)}$. It can be shown that

$$\Theta_{\ell,r}(R') \ge \Theta_{\ell,r-1}(R') \quad \Longleftrightarrow \quad R' \ge \Upsilon_{\ell,r} \,.$$

$\Longrightarrow$ Optimal r is the unique integer satisfying
$\Upsilon_{\ell,r} \le R' < \Upsilon_{\ell,r+1}$ (best r is a function of R' and ℓ).

- We have:

$$\Theta_\ell(R') = \begin{cases} \Theta_{\ell,1}(R') & \Upsilon_{\ell,1} \le R' < \Upsilon_{\ell,2} \\ \Theta_{\ell,2}(R') & \Upsilon_{\ell,2} \le R' < \Upsilon_{\ell,3} \\ \vdots & \vdots \\ \Theta_{\ell,\ell}(R') & \Upsilon_{\ell,\ell} \le R' < \Upsilon_{\ell,\ell+1} \end{cases}$$

$(\Upsilon_{\ell,1} \triangleq 0,\ \Upsilon_{\ell,\ell+1} \triangleq 1)$

Optimizing the decoding radius

- We can optimize over r to obtain the best possible decoding radius for the GS decoder. Define

$$\Theta_\ell(R') = \max_{1 \le r \le \ell} \Theta_{\ell,r}(R') \,.$$

- Define $\Upsilon_{\ell,r} = \frac{r(r-1)}{\ell(\ell+1)}$. It can be shown that

$$\Theta_{\ell,r}(R') \ge \Theta_{\ell,r-1}(R') \quad \Longleftrightarrow \quad R' \ge \Upsilon_{\ell,r} \,.$$

$\Longrightarrow$ Optimal r is the unique integer satisfying
$\Upsilon_{\ell,r} \le R' < \Upsilon_{\ell,r+1}$ (best r is a function of R' and ℓ).

- We have:

$$\Theta_\ell(R') = \begin{cases} \Theta_{\ell,1}(R') & \Upsilon_{\ell,1} \le R' < \Upsilon_{\ell,2} \\ \Theta_{\ell,2}(R') & \Upsilon_{\ell,2} \le R' < \Upsilon_{\ell,3} \\ \vdots & \vdots \\ \Theta_{\ell,\ell}(R') & \Upsilon_{\ell,\ell} \le R' < \Upsilon_{\ell,\ell+1} \end{cases}$$

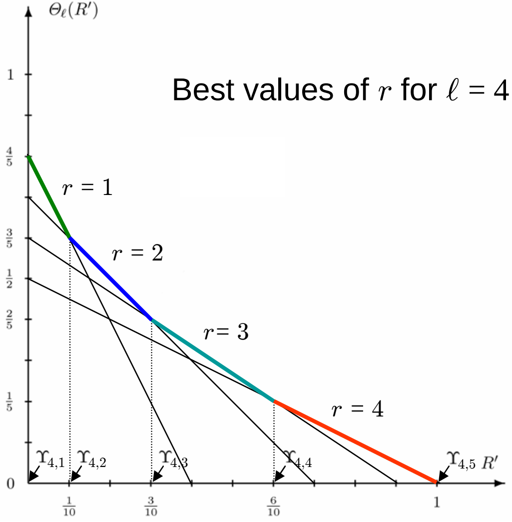
$(\Upsilon_{\ell,1} \triangleq 0\,,\ \Upsilon_{\ell,\ell+1} \triangleq 1)$

Asymptotics

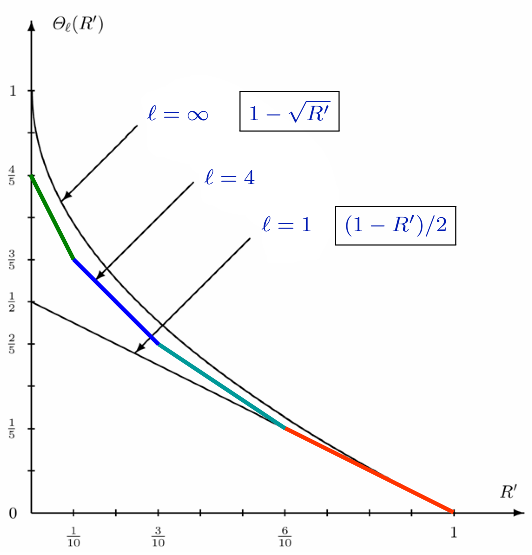
The value of $\Theta_\ell(R')$ is always non-decreasing with ℓ, and it can be shown that

$$\Theta_\infty(R') \triangleq \lim_{\ell \to \infty} \Theta_\ell(R') = 1 - \sqrt{R'} \,.$$

Best values of r for $\ell = 4$



Comparison with list-1 decoder and asymptotic behavior



The average number of incorrect codewords in the list

It turns out that in *most cases* the list produced by the GS decoder contains *just one codeword* (the closest codeword to the received word).

The average number of incorrect codewords in the list

It turns out that in *most cases* the list produced by the GS decoder contains *just one codeword* (the closest codeword to the received word).

- McEliece (2003) shows that under a q-ary symmetric channel ($q = |F|$), the average number of "bad" codewords in the list produced by a GS decoder of radius τ is very close to

$$\bar{L}(\tau) = q^{-(n-k)} \sum_{s=0}^{\tau} \binom{n}{s} (q-1)^s .$$

average number of codewords in a *random* sphere of radius τ

The average number of incorrect codewords in the list

It turns out that in *most cases* the list produced by the GS decoder contains *just one codeword* (the closest codeword to the received word).

- McEliece (2003) shows that under a q-ary symmetric channel ($q = |F|$), the average number of "bad" codewords in the list produced by a GS decoder of radius τ is very close to

$$\bar{L}(\tau) = q^{-(n-k)} \sum_{s=0}^{\tau} \binom{n}{s} (q-1)^s \,.$$

average number of codewords in a *random* sphere of radius τ

- Ruckenstein (Ph.D. Thesis, 2001) gave the explicit estimate

$$\bar{L}_{\mathsf{bad}} \le q^{-\varepsilon n} \quad \text{whenever} \quad \sqrt{k/n} - k/n - 1/\log_2 q \ge \varepsilon \,.$$

Example

Consider a $[256, 179]$ GRS code. We have $R = k/n \approx 0.7$, and thus

$$\bar{L}_{\mathsf{bad}} \approx 256^{-(\sqrt{0.7}-0.7-0.125)\cdot 256} \approx 6.5 \times 10^{-8},$$

with $\tau \approx 41$ (conventional list-1 decoder corrects 38 errors).

Finding z-roots of bi-variate polynomials

- The goal: given $Q(x,z) \in F[x,z]$, and an integer $k > 0$, find all factors of $Q(x,z)$ of the form $z - u(x)$, with $u(x) \in F[x]$ and $\deg u(x) < k$.

Finding z-roots of bi-variate polynomials

- The goal: given $Q(x,z) \in F[x,z]$, and an integer $k > 0$, find all factors of $Q(x,z)$ of the form $z - u(x)$, with $u(x) \in F[x]$ and $\deg u(x) < k$.
- The observation: if

$$\left(z - u_0 - u_1 x - \cdots - u_{k-1}x^{k-1}\right) \Bigg| \; Q(x,z) \text{ and } x \nmid Q(x,z)$$

  then $(z - u_0) \mid Q(0,z) \implies u_0$ is a root of $Q(0,z) \in F[z]$.
  - Find u_0 using a root-finding algorithm for univariate polynomials. For example, Chien search is $O(|F|)$, which is $O(n)$ when $n \approx |F|$ (e.g., primitive RS codes). More sophisticated methods exist.

Finding z-roots of bi-variate polynomials

- The goal: given $Q(x,z) \in F[x,z]$, and an integer $k > 0$, find all factors of $Q(x,z)$ of the form $z - u(x)$, with $u(x) \in F[x]$ and $\deg u(x) < k$.
- The observation: if

$$(z - u_0 - u_1 x - \cdots - u_{k-1}x^{k-1}) \Big| \, Q(x,z) \text{ and } x \nmid Q(x,z)$$

then $(z - u_0) \mid Q(0,z) \implies u_0$ is a root of $Q(0,z) \in F[z]$.

  - Find u_0 using a root-finding algorithm for univariate polynomials. For example, Chien search is $O(|F|)$, which is $O(n)$ when $n \approx |F|$ (e.g., primitive RS codes). More sophisticated methods exist.

- Let $z' = zx + u_0$. Then,

$$z' - u(x) = zx - u_1 x - u_2 x^2 - \cdots - u_{k-1}x^{k-1} = x(z - u_1 - u_2 x - \cdots - u_{k-1}x^{k-2})$$

and we get that

$$(z - u_1 - u_2 x - \cdots - u_{k-1}x^{k-2}) \Big| \, x^{-1}Q(x, xz + u_0)\,.$$

We proceed recursively, recovering $u_0, u_1, \ldots, u_{k-1}$.

Algoritm BiRoot (Roth-Ruckenstein 2000)

$\textsc{BiRoot}\,(\,Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}\,)$

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

if ($\lambda == 0$)
    $U \leftarrow \emptyset$;
$m \leftarrow$ largest integer such that $x^m \mid Q(x,z)$;
$T(x,z) \leftarrow x^{-m} Q(x,z)$;
$Z \leftarrow$ set of all distinct (z-)roots of $T(0,z)$ in F;
for each $\gamma \in Z$ do $\{$
    $g_\lambda \leftarrow \gamma$;
    if ($\lambda < k-1$)
        $\textsc{BiRoot}(T(x, xz+\gamma), k, \lambda+1)$;
    else
        if ($Q(x, g_{k-1}) == 0$)
            $U \leftarrow U \cup \{g(x)\}$;
$\}$

Algoritm BiRoot (Roth-Ruckenstein 2000)

BiRoot ($Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}$)

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

```
if (λ == 0)
    U ← ∅;
m ← largest integer such that x^m | Q(x,z);
T(x,z) ← x^(−m) Q(x,z);
Z ← set of all distinct (z-)roots of T(0,z) in F;
for each γ ∈ Z do {
    g_λ ← γ;
    if ( λ < k−1 )
        BiRoot(T(x, xz + γ), k, λ+1);
    else
        if ( Q(x, g_{k−1}) == 0 )
            U ← U ∪ {g(x)};
}
```

Algoritm BiRoot (Roth-Ruckenstein 2000)

BiRoot $(\, Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}\,)$

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

if $(\lambda == 0)$
    $U \leftarrow \emptyset$;
$m \leftarrow$ largest integer such that $x^m \mid Q(x,z)$;
$T(x,z) \leftarrow x^{-m} Q(x,z)$;
$Z \leftarrow$ set of all distinct $(z$-$)$roots of $T(0,z)$ in F;
for each $\gamma \in Z$ do $\{$
    $g_\lambda \leftarrow \gamma$;
    if $(\, \lambda < k-1 \,)$
        $\textsc{BiRoot}(T(x, xz+\gamma), k, \lambda+1)$;
    else
        if $(\, Q(x, g_{k-1}) == 0 \,)$
            $U \leftarrow U \cup \{g(x)\}$;
$\}$

Algoritm BiRoot (Roth-Ruckenstein 2000)

$\textsc{BiRoot}\,(\,Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}\,)$

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

```
if (λ == 0)
    U ← ∅;
m ← largest integer such that x^m | Q(x,z);
T(x,z) ← x^(−m) Q(x,z);
Z ← set of all distinct (z-)roots of T(0,z) in F;
for each γ ∈ Z do {
    g_λ ← γ;
    if (λ < k−1)
        BiRoot(T(x, xz + γ), k, λ+1);
    else
        if (Q(x, g_{k−1}) == 0)
            U ← U ∪ {g(x)};
}
```

Algoritm BiRoot (Roth-Ruckenstein 2000)

BIROOT ($Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}$)

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

if ($\lambda == 0$)
    $U \leftarrow \emptyset$;
$m \leftarrow$ largest integer such that $x^m \mid Q(x,z)$;
$T(x,z) \leftarrow x^{-m} Q(x,z)$;
$Z \leftarrow$ set of all distinct (z-)roots of $T(0,z)$ in F;
for each $\gamma \in Z$ do {
    $g_\lambda \leftarrow \gamma$;
    if ($\lambda < k-1$)
        $\textsc{BiRoot}(T(x, xz+\gamma), k, \lambda+1)$;
    else
        if ($Q(x, g_{k-1}) == 0$)
            $U \leftarrow U \cup \{g(x)\}$;
}

Algoritm BiRoot (Roth-Ruckenstein 2000)

BiRoot $(\, Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}\,)$

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

if $(\lambda == 0)$
    $U \leftarrow \emptyset$;
$m \leftarrow$ largest integer such that $x^m \mid Q(x,z)$;
$T(x,z) \leftarrow x^{-m} Q(x,z)$;
$Z \leftarrow$ set of all distinct (z-)roots of $T(0,z)$ in F;
for each $\gamma \in Z$ do $\{$
    $g_\lambda \leftarrow \gamma$;
    if $(\,\lambda < k-1\,)$
        $\mathrm{BiRoot}(T(x, xz+\gamma), k, \lambda+1)$;
    else
        if $(\, Q(x, g_{k-1}) == 0\,)$
            $U \leftarrow U \cup \{g(x)\}$;
$\}$

Algoritm BiRoot (Roth-Ruckenstein 2000)

$\textsc{BiRoot}\,(\,Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}\,)$

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

if $(\lambda == 0)$
    $U \leftarrow \emptyset$;
$m \leftarrow$ largest integer such that $x^m \mid Q(x,z)$;
$T(x,z) \leftarrow x^{-m} Q(x,z)$;
$Z \leftarrow$ set of all distinct (z-)roots of $T(0,z)$ in F;
for each $\gamma \in Z$ do $\{$
    $g_\lambda \leftarrow \gamma$;
    if $(\,\lambda < k{-}1\,)$
        $\textsc{BiRoot}(T(x, xz+\gamma), k, \lambda{+}1)$;
    else
        if $(\,Q(x, g_{k-1}) == 0\,)$
            $U \leftarrow U \cup \{g(x)\}$;
$\}$

Algoritm BiRoot (Roth-Ruckenstein 2000)

$\textsc{BiRoot}\,(\,Q(x,z) \in F[x,y],\ k \in \mathbb{N},\ \lambda \in \mathbb{N}\,)$

// Input λ is the recursion depth.
// Global variables: set $U \subseteq F_k[x]$, polynomial $g(x) = \sum_{s=0}^{k-1} g_s x^s \in F_k[x]$.
// On output, U contains *all* z-linear factors of $Q(x,z)$.
// Call procedure initially with $Q(x,z) \neq 0$, $k > 0$, and $\lambda = 0$.

if ($\lambda == 0$)
    $U \leftarrow \emptyset$;
$m \leftarrow$ largest integer such that $x^m \mid Q(x,z)$;
$T(x,z) \leftarrow x^{-m} Q(x,z)$;
$Z \leftarrow$ set of all distinct (z-)roots of $T(0,z)$ in F;
for each $\gamma \in Z$ do $\{$
    $g_\lambda \leftarrow \gamma$;
    if ($\lambda < k{-}1$)
        $\textsc{BiRoot}(T(x, xz+\gamma), k, \lambda{+}1)$;
    else
        if ($Q(x, g_{k-1}) == 0$)
            $U \leftarrow U \cup \{g(x)\}$;
$\}$

Algorithm BiRoot: correctness

Proposition

Let $Q(x, z)$ be a nonzero bivariate polynomial in $F[x, z]$ and let U be the set that is computed by the call $\textsc{BiRoot}(Q, k, 0)$. Then, every element of U is a z-root of $Q(x, z)$, and every z-root of $Q(x, z)$ is contained in U.

Proof: Roth & Ruckenstein (2000), Roth (2005).

Algorithm BiRoot: complexity

- The z-degree of $Q_i(x, z)$ and $T(x, z)$ does not change during execution
  $\Longrightarrow$ $T(0, z)$ in Step // 5 // is nonzero and of finite, bounded degree
  $\Longrightarrow$ Step // 5 // returns a finite set.
- Clearly, the recursion depth is limited to k in Step // 8 //
  $\Longrightarrow$ BiRoot terminates.

Algorithm BiRoot: complexity

- The z-degree of $Q_i(x,z)$ and $T(x,z)$ does not change during execution
  $\Longrightarrow$ $T(0,z)$ in Step // 5 // is nonzero and of finite, bounded degree
  $\Longrightarrow$ Step // 5 // returns a finite set.
- Clearly, the recursion depth is limited to k in Step // 8 //
  $\Longrightarrow$ BiRoot terminates.
- Roth (2005) shows that if the z-degree of $Q(x,z)$ is ℓ then the total number of recursive calls made to BiRoot is at most $\ell(k-1)$
  $\Longrightarrow$ BiRoot runs in polynomial time if the root finder of Step // 5 // does.

Algorithm BiRoot: complexity

- The z-degree of $Q_i(x,z)$ and $T(x,z)$ does not change during execution
  $\Longrightarrow$ $T(0,z)$ in Step //5// is nonzero and of finite, bounded degree
  $\Longrightarrow$ Step //5// returns a finite set.
- Clearly, the recursion depth is limited to k in Step //8//
  $\Longrightarrow$ BiRoot terminates.
- Roth (2005) shows that if the z-degree of $Q(x,z)$ is ℓ then the total number of recursive calls made to BiRoot is at most $\ell(k-1)$
  $\Longrightarrow$ BiRoot runs in polynomial time if the root finder of Step //5// does.

Detailed complexity analysis can be found in Roth (2005), and Roth & Ruckenstein (2000). Assuming complexity $O(\ell^2 \log^2 \ell \log |F|)$ for root-finding in $F[z]$, the total complexity of BiRoot is $O((\ell \log^2 \ell) k(n + \ell \log |F|))$.