6. Interpolation from erroneous data

CS-E4500 Advanced Course on Algorithms
Spring 2020

Petteri Kaski
Department of Computer Science
Aalto University

Lecture schedule

Tue 7 Jan:	1. Polynomials and integers
Tue 14 Jan:	2. The fast Fourier transform and fast multiplication
Tue 21 Jan:	3. Quotient and remainder
Tue 28 Jan:	4. Batch evaluation and interpolation
Tue 4 Feb:	5. Extended Euclidean algorithm
Tue 11 Feb:	*Break — no lecture*
Tue 18 Feb:	*Exam week — no lecture*
Tue 26 Feb:	6. Interpolation from erroneous data
Tue 3 Mar:	7. Factor graphs and contraction
Tue 10 Mar:	8. Verifiable and error-tolerant inference for factor graphs
Tue 17 Mar:	9. Further applications of factor graphs

CS-E4500 Advanced Course in Algorithms (5 ECTS, III–IV, Spring 2020)

2020 KALENTERI 2020

Tammikuu

1	Ke	Uudenvuodenpäivä
2	To	
3	Pe	
4	La	
5	Su	
6	Ma	Loppiainen
7	Ti	L1
8	Ke	
9	To	Q1
10	Pe	
11	La	
12	Su	D1
13	Ma	Vk 03 T1
14	Ti	L2
15	Ke	
16	To	Q2
17	Pe	
18	La	
19	Su	D2
20	Ma	Vk 04 T2
21	Ti	L3
22	Ke	
23	To	Q3
24	Pe	
25	La	
26	Su	D3
27	Ma	Vk 05 T3
28	Ti	L4
29	Ke	
30	To	Q4
31	Pe	

Helmikuu

1	La	
2	Su	D4
3	Ma	Vk 06 T4
4	Ti	L5
5	Ke	
6	To	Q5
7	Pe	
8	La	
9	Su	
10	Ma	Vk 07
11	Ti	Break
12	Ke	week
13	To	
14	Pe	
15	La	
16	Su	
17	Ma	Vk 08
18	Ti	Exam
19	Ke	week
20	To	
21	Pe	
22	La	
23	Su	D5
24	Ma	Vk 09 T5
25	Ti	Laskiainen
26	Ke	
27	To	Q6
28	Pe	
29	La	

Maaliskuu

1	Su	D6
2	Ma	Vk 10 T6
3	Ti	L7
4	Ke	
5	To	Q7
6	Pe	
7	La	
8	Su	D7
9	Ma	Vk 11 T7
10	Ti	L8
11	Ke	
12	To	Q8
13	Pe	
14	La	
15	Su	D8
16	Ma	Vk 12 T8
17	Ti	L9
18	Ke	
19	To	Q9
20	Pe	Kevätpäiväntasaus
21	La	
22	Su	D9
23	Ma	Vk 13 T9
24	Ti	
25	Ke	
26	To	
27	Pe	
28	La	
29	Su	Kesäaika alkaa
30	Ma	Vk 14
31	Ti	

Huhtikuu

1	Ke	
2	To	
3	Pe	
4	La	
5	Su	Palmusunnuntai
6	Ma	Vk 15
7	Ti	
8	Ke	
9	To	
10	Pe	Pitkäperjantai
11	La	
12	Su	Pääsiäispäivä
13	Ma	2. pääsiäispäivä
14	Ti	
15	Ke	
16	To	
17	Pe	
18	La	
19	Su	
20	Ma	Vk 17
21	Ti	
22	Ke	
23	To	
24	Pe	
25	La	
26	Su	
27	Ma	Vk 18
28	Ti	
29	Ke	
30	To	

Toukokuu

1	Pe	Vappu
2	La	
3	Su	
4	Ma	Vk 19
5	Ti	
6	Ke	
7	To	
8	Pe	
9	La	
10	Su	Äitienpäivä
11	Ma	Vk 20
12	Ti	
13	Ke	
14	To	
15	Pe	
16	La	
17	Su	Kaatuneiden muistopäivä
18	Ma	Vk 21
19	Ti	
20	Ke	
21	To	Helatorstai
22	Pe	
23	La	
24	Su	
25	Ma	Vk 22
26	Ti	
27	Ke	
28	To	
29	Pe	
30	La	
31	Su	Helluntaipäivä

Kesäkuu

1	Ma	Vk 23
2	Ti	
3	Ke	
4	To	
5	Pe	
6	La	
7	Su	
8	Ma	Vk 24
9	Ti	
10	Ke	
11	To	
12	Pe	
13	La	
14	Su	
15	Ma	Vk 25
16	Ti	
17	Ke	
18	To	
19	Pe	
20	La	Juhannus
21	Su	Kesäpäivänseisaus
22	Ma	Vk 26
23	Ti	
24	Ke	
25	To	
26	Pe	
27	La	
28	Su	
29	Ma	Vk 27
30	Ti	

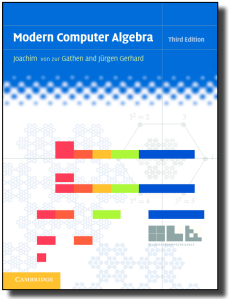
L = Lecture; hall T5, Tue 12–14
Q = Q & A session; hall T5, Thu 12–14
D = Problem set deadline; Sun 20:00
T = Tutorial (model solutions); hall T6, Mon 16–18

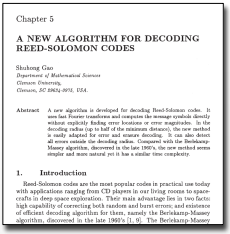
Recap of last week (before the break)

- **Extended Euclidean algorithm** for polynomials recalled and expanded
  - The **quotient sequence**, the **Bézout coefficients**, and the **halting threshold**
- Fast extended Euclidean algorithm for polynomials by **divide and conquer**
  - The two polynomial operands **truncated** to a prefix of the highest-degree monomials determine the prefix of the quotient sequence (exercise)

Goal: Near-linear-time toolbox for univariate polynomials

- Multiplication
- Division (quotient and remainder)
- Batch evaluation
- Interpolation
- Extended Euclidean algorithm (gcd)
- Interpolation from partly erroneous data (this week)





Chapter 5

A NEW ALGORITHM FOR DECODING REED-SOLOMON CODES

Shuhong Gao
Department of Mathematical Sciences
Clemson University,
Clemson, SC 29634-1907, USA.

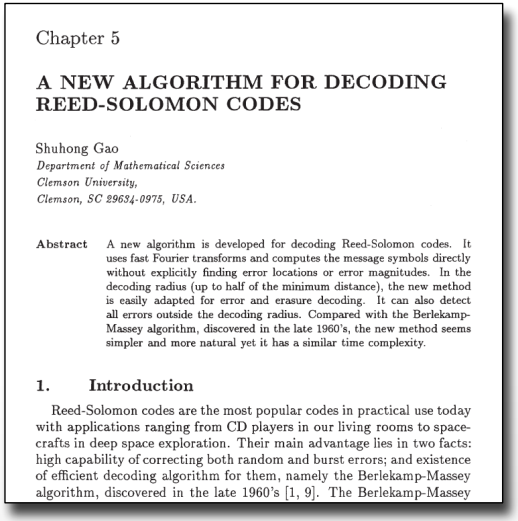
Abstract: A new algorithm is developed for decoding Reed-Solomon codes. It uses fast Fourier transforms and computes the message symbols directly without explicitly finding error locations or error magnitudes. In the decoding radius (up to half of the minimum distance), the new method is easily adapted for error and erasure decoding. It can also detect all errors outside the decoding radius. Compared with the Berlekamp-Massey algorithm, discovered in the late 1960's, the new method seems simpler and more natural yet it has a similar time complexity.

1. Introduction

Reed-Solomon codes are the most popular codes in practical use today with applications ranging from CD players in our living rooms to spacecrafts in deep space exploration. Their main advantage lies in two facts: high capability of correcting both random and burst errors; and existence of efficient decoding algorithm for them, namely the Berlekamp-Massey algorithm, discovered in the late 1960's [1, 9]. The Berlekamp-Massey

Fast interpolation from partly erroneous data

(Gao [5])



Chapter 5

A NEW ALGORITHM FOR DECODING REED-SOLOMON CODES

Shuhong Gao
Department of Mathematical Sciences
Clemson University,
Clemson, SC 29634-0975, USA.

Abstract A new algorithm is developed for decoding Reed-Solomon codes. It uses fast Fourier transforms and computes the message symbols directly without explicitly finding error locations or error magnitudes. In the decoding radius (up to half of the minimum distance), the new method is easily adapted for error and erasure decoding. It can also detect all errors outside the decoding radius. Compared with the Berlekamp-Massey algorithm, discovered in the late 1960's, the new method seems simpler and more natural yet it has a similar time complexity.

1. Introduction

Reed-Solomon codes are the most popular codes in practical use today with applications ranging from CD players in our living rooms to spacecrafts in deep space exploration. Their main advantage lies in two facts: high capability of correcting both random and burst errors; and existence of efficient decoding algorithm for them, namely the Berlekamp-Massey algorithm, discovered in the late 1960's [1, 9]. The Berlekamp-Massey

Further motivation for this week

- After this week we have completed our work on the near-linear time toolbox for univariate polynomials
- This week is also our first encounter with **uncertainty** in computation
- This week we learn how to cope with uncertainty in the form of **errors in data** by using **error-correcting codes**
- In subsequent weeks we look at **errors in computation** ...

Key content for Lecture 6

- Coping with **errors in data** using **error-correcting codes**
- A family of error-correcting codes (**Reed–Solomon codes**) based on evaluation–interpolation duality for univariate polynomials
  - Key observation: low-degree polynomials have few roots (exercise)
  - Fast **encoding** and **decoding** of Reed–Solomon codes via the fast univariate polynomial toolkit and **Gao's** (2003) **decoder**

Number of roots

- Let F be a field
- A **root** of a polynomial $f \in F[x]$ is an element $\xi \in F$ with $f(\xi) = 0$

Theorem 10 (Number of roots)

A nonzero polynomial $f \in F[x]$ of degree at most d has at most d distinct roots.

Proof.

Exercise

□

Two distinct polynomials mostly disagree

- Let F be a field
- Let $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) \in F^e$ be a vector of e distinct elements of F
- Associate with $f \in F[x]$ the vector of evaluations

$$f(\Xi) = (f(\xi_1), f(\xi_2), \ldots, f(\xi_e)) \in F^e$$

Lemma 11 (Bounded agreement of low-degree polynomials)

Let $f_0, f_1 \in F[x]$ be distinct polynomials of degree at most d.
Then, $f_0(\Xi)$ and $f_1(\Xi)$ agree in at most d coordinates.

Proof.

The difference $f_0 - f_1 \neq 0$ is a polynomial of degree at most d and thus has at most d distinct roots

□

Reconstructibility from partly erroneous data

- Let $f \in F[x]$ be a polynomial of degree at most d
- Let $e \geq d+1$ and let $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) \in F^e$ consist of distinct elements

Lemma 12 (Unique reconstructibility)

Suppose that the vectors $\Gamma \in F^e$ and $f(\Xi)$ disagree in at most $(e-d-1)/2$ coordinates. Then, Γ uniquely identifies f

Proof.

Let $f_0, f_1 \in F[x]$ be two polynomials of degree at most d such that $f_0(\Xi)$ and $f_1(\Xi)$ each disagree with Γ in at most $(e-d-1)/2$ coordinates. In total there are e coordinates, so $f_0(\Xi)$ and $f_1(\Xi)$ and Γ must thus all agree in at least $e - 2(e-d-1)/2 = d+1$ coordinates. By Lemma 11 thus $f_0 = f_1$. □

(Furthermore, we can, very inefficiently, recover f from Γ by considering in turn each vector $\tilde{\Gamma} \in F^e$ that disagrees with Γ in at most $(e-d-1)/2$ coordinates: for each such $\tilde{\Gamma}$, interpolate f from $f(\Xi) = \tilde{\Gamma}$, and stop when f has degree at most d.)

Reed–Solomon codes

- Suppose we want to protect a sequence $\Phi = (\varphi_0, \varphi_1, \ldots, \varphi_d) \in F^{d+1}$ of elements of a field F against errors
- We may represent Φ as a polynomial $f = \varphi_0 + \varphi_1 x + \ldots + \varphi_d x^d \in F[x]$ of degree at most d
- Let $e \geq d + 1$ and let $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) \in F^e$ consist of distinct elements
- Let us use $\Psi = f(\Xi) \in F^e$ as the encoded representation of Φ
- Suppose that $\hat{\Psi}$ disagrees with Ψ in at most $(e - d - 1)/2$ coordinates. Then, Lemma 12 implies that we can recover Φ from $\hat{\Psi}$
- That is, $\hat{\Psi}$ may have up to $\lfloor (e - d - 1)/2 \rfloor$ errors and we can still recover Φ
- Encoding can be done in near-linear-time by fast batch evaluation ...
- ... but how efficiently can we decode in the presence of errors?

Example: Encoding

- Let us work with $e = 8$, $d = 3$, $F = \mathbb{Z}_{11}$, and the evaluation points $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) = (0, 1, 2, 3, 4, 5, 6, 7) \in \mathbb{Z}_{11}^e$
- Suppose we want to protect the data vector $\Phi = (5, 3, 1, 9) \in \mathbb{Z}_{11}^{d+1}$
- We view Φ as the degree-at-most-d polynomial $f = 5 + 3x + x^2 + 9x^3 \in \mathbb{Z}_{11}[x]$
- The encoded representation of Φ is

$$\Psi = f(\Xi) = (f(\xi_1), f(\xi_2), \ldots, f(\xi_e)) = (5, 7, 10, 2, 4, 4, 1, 5) \in \mathbb{Z}_{11}^e$$

Gao's (2003) decoder for Reed–Solomon codes

- Let $f \in F[x]$ be a polynomial of degree at most d
- Let $e \geq d + 1$ and let $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) \in F^e$ consist of distinct elements
- Suppose that the vectors $\Gamma \in F^e$ and $f(\Xi)$ disagree in at most $(e - d - 1)/2$ coordinates. Then, Γ uniquely identifies f (Lemma 12)
- Moreover, given Ξ, Γ, d as input, f can be computed in $O(M(e) \log e)$ operations in F (Gao [5])

Gao's decoding algorithm

- Let $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) \in F^e$ consisting of distinct elements, $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_e) \in F^e$, and $d \in \mathbb{Z}_{\geq 0}$ with $d + 1 \leq e$ be given as input
- Gao's algorithm [5] proceeds as follows:
  1. Using a subproduct tree, construct the polynomial $g_0 = \prod_{i=1}^{e}(x - \xi_i)$
  2. Interpolate the unique polynomial $g_1 \in F[x]$ of degree at most $e - 1$ that satisfies $g_1(\xi_i) = \gamma_i$ for all $i = 1, 2, \ldots, e$
  3. Apply the extended Euclidean algorithm to g_0 and g_1 to produce the consecutive remainders g_h, g_{h+1} with $\deg g_h \geq D$, and $\deg g_{h+1} < D$ for $D = (e + d + 1)/2$. Let $s_{h+1}, t_{h+1} \in F[x]$ be the associated Bézout coefficients with $g_{h+1} = s_{h+1}g_0 + t_{h+1}g_1$
  4. Divide g_{h+1} by t_{h+1} to obtain the quotient $f_1 \in F[x]$ and the remainder $r \in F[x]$ with $g_{h+1} = t_{h+1}f_1 + r$ and $\deg r < \deg t_{h+1}$
  5. Output f_1 as the result of interpolation if both $\deg f_1 \leq d$ and $r = 0$; otherwise assert decoding failure
- It is immediate that the algorithm runs in $O(M(e) \log e)$ operations in F

Example: Decoding I

- Let us work with $e = 8$, $d = 3$, $F = \mathbb{Z}_{11}$, and the evaluation points $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) = (0, 1, 2, 3, 4, 5, 6, 7) \in \mathbb{Z}_{11}^e$
- Suppose we have the vector $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_e) = (5, 7, 1, 2, 9, 4, 1, 5) \in \mathbb{Z}_{11}^e$
- First, we construct the polynomial

$$g_0 = \prod_{i=1}^{e} (x - \xi_i) = 9x + 2x^3 + 4x^4 + 9x^5 + 3x^6 + 5x^7 + x^8$$

- Then, we interpolate the polynomial

$$g_1 = 5 + 7x + 5x^2 + 2x^3 + 10x^4 + 9x^5 + 6x^6 + 7x^7$$

that satisfies $g_1(\xi_i) = \gamma_i$ for all $i = 1, 2, \ldots, e$

Example: Decoding II

- Next we apply the extended Euclidean algorithm to g_0 and g_1 to produce the consecutive remainders g_h, g_{h+1} with $\deg g_h \geq D$, and $\deg g_{h+1} < D$ for $D = (e + d + 1)/2 = 6$ ...
- For convenience, we display the entire output of the extended Euclidean algorithm (but omitting the first Bézout coefficient sequence):

i	q_i	g_i	t_i
0		$9x + 2x^3 + 4x^4 + 9x^5 + 3x^6 + 5x^7 + x^8$	0
1	$8 + 8x$	$5 + 7x + 5x^2 + 2x^3 + 10x^4 + 9x^5 + 6x^6 + 7x^7$	1
2	$7 + 10x$	$4 + x + 3x^2 + x^3 + 7x^4 + 4x^6$	$3 + 3x$
3	$3 + 3x$	$10 + 4x + 7x^2 + 9x^3 + 6x^4 + 5x^5$	$2 + 4x + 3x^2$
4	$6 + 10x$	$7 + 3x + 3x^2 + 8x^3 + 6x^4$	$8 + 7x + x^2 + 2x^3$
5	$10 + 9x$	$1 + 4x + 3x^2 + 8x^3$	$9 + 3x + 4x^2 + 2x^4$
6	$4 + 10x$	$8 + 9x + 3x^2$	$6 + 6x + 10x^3 + 2x^4 + 4x^5$
7	$5 + 4x$	$2 + 9x$	$7 + 7x + 10x^2 + 4x^3 + 4x^4 + 8x^5 + 4x^6$
8	$10 + x$	9	$4 + 9x + 10x^2 + 5x^3 + 10x^4 + 3x^5 + 3x^6 + 6x^7$
9		0	$x + 10x^3 + 9x^4 + x^5 + 4x^6 + 3x^7 + 5x^8$

- (In a fast implementation, we would of course use the divide-and-conquer extended Euclidean algoritm and would not produce the entire sequence of remainders g_i)

Example: Decoding III

- From the extended Euclidean algorithm we obtain that $h = 2$ with

$$g_{h+1} = 10 + 4x + 7x^2 + 9x^3 + 6x^4 + 5x^5$$
$$t_{h+1} = 2 + 4x + 3x^2$$

- Dividing g_{h+1} by t_{h+1} we obtain the quotient

$$f_1 = 5 + 3x + x^2 + 9x^3$$

  and the remainder $r = 0$
- In particular, the decoding is successful, and the reconstructed data vector is $(5, 3, 1, 9) \in \mathbb{Z}_{11}^{d+1}$
- Re-encoding the reconstructed vector as appropriate, we can also observe that the vector Γ has two errors, namely $f(\xi_3) = 10 \neq \gamma_3 = 2$ and $f(\xi_5) = 4 \neq \gamma_5 = 9$

Correctness I

- First, suppose that the algorithm does not assert failure
- Then, $f_1 = g_{h+1}/t_{h+1}$ has degree at most d
- Since $t_{h+1}f_1 = g_{h+1} = s_{h+1}g_0 + t_{h+1}g_1$, we have $s_{h+1}g_0 = t_{h+1}(f_1 - g_1)$ and hence for all $i = 1, 2, \ldots, e$ we have $t_{h+1}(\xi_i) = 0$ or $f_1(\xi_i) = g_1(\xi_i) = \gamma_i$
- Since g_{h+1} is the first remainder with $\deg g_{h+1} < D$ and $\deg g_0 = e$, by the structure of the Bézout coefficients we have $\deg t_{h+1} \leq e - D = (e - d - 1)/2$
- Indeed, from the definition of Bézout coefficients we have $\deg s_{h+1}, \deg t_{h+1} \leq \sum_{i=1}^{h} \deg q_i = \deg g_0 - \deg g_h \leq e - D$ since $\deg g_i + \deg q_i = \deg g_{i-1}$ and $\deg g_h \geq D$
- Since t_{h+1} has at most $\deg t_{h+1}$ roots, we have $f_1(\xi_i) \neq \gamma_i$ for at most $(e - d - 1)/2$ coordinates $i = 1, 2, \ldots, e$
- Thus, f_1 is a valid output for input Ξ, Γ, d

Correctness II

- Next, let $f \in F[x]$ be a polynomial of degree at most d, let $\Xi = (\xi_1, \xi_2, \ldots, \xi_e) \in F^e$ consist of distinct elements, and let $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_e) \in F^e$ be a vector that disagrees with $f(\Xi)$ in at most $(e - d - 1)/2$ coordinates for $d + 1 \leq e$
- By Lemma 12, we know that Γ uniquely determines f
- We show that Gao's algorithm outputs $f_1 = f$ on input Ξ, Γ, d
- Let $B = \{i \in \{1, 2, \ldots, e\} : f(\xi_i) \neq \gamma_i\}$ be the set of "bad" coordinates
- That is, B is the set of coordinates where Γ and $f(\Xi)$ disagree
- By assumption we have $|B| \leq (e - d - 1)/2$
- To understand the operation of the algorithm, let us split the polynomials g_0 and g_1 into parts based on B and $G = \{1, 2, \ldots, e\} \setminus B$ (the "bad" and "good" coordinates)

Correctness III

- Toward this end, let

$$q = \prod_{i \in G} (x - \xi_i) \in F[x], \qquad r_0 = \prod_{i \in B} (x - \xi_i) \in F[x]$$

- It is immediate that $g_0 = qr_0$
- Let $r_1 \in F[x]$ be the unique polynomial of degree at most $(e - d - 1)/2 - 1$ with $r_1(\xi_i) = q(\xi_i)^{-1}(\gamma_i - f(\xi_i)) \neq 0$ for all $i \in B$
- Thus, we have $g_1 = qr_1 + f$
- We have that $\gcd(r_0, r_1) = 1$ since no root of r_0 is a root of r_1 and r_0 factors into a product of degree 1 polynomials
- The following lemma will imply that the algorithm outputs $f_1 = f$; we postpone the proof and give it as Lemma 13

Correctness IV

- **Gao's Lemma.** (Lemma 13 below) Let $c, d, D \in \mathbb{Z}_{\geq 0}$ and let $q, r_0, r_1, f_0, f_1 \in F[x]$ with $\gcd(r_0, r_1) = 1$, $\deg q \geq D \geq c + d + 1$, and $\deg r_i \leq c$, $\deg f_i \leq d$ for $i = 0, 1$. Run the extended Euclidean algorithm on input $g_0 = qr_0 + f_0$ and $g_1 = qr_1 + f_1$ to obtain the remainders g_h and $g_{h+1} = s_{h+1}g_0 + t_{h+1}g_1$ for $s_{h+1}, t_{h+1} \in F[x]$ with $\deg g_h \geq D$ and $\deg g_{h+1} < D$. Then, $s_{h+1} = -\alpha r_1$ and $t_{h+1} = \alpha r_0$ for some $\alpha \in F \setminus \{0\}$
- Take $f_0 = 0$, $f_1 = f$, $c = |B|$ in the lemma and recall that we have $D = (e + d + 1)/2$
- Thus, $c \leq (e - d - 1)/2$, $\deg q = |G| = e - |B| \geq D \geq c + d + 1$, and the lemma applies to the polynomials $g_0 = qr_0$ and $g_1 = qr_1 + f$ constructed in the algorithm
- Let $g_{h+1}, s_{h+1}, t_{h+1}$ be the output of the lemma (also constructed by the algorithm)
- Because $f_0 = 0$ and $f_1 = f$, we have $g_{h+1} = -\alpha r_1 q r_0 + \alpha r_0(qr_1 + f) = t_{h+1}f$
- In particular, the algorithm outputs $f_1 = f = g_{h+1}/t_{h+1}$ $\square$

Preparation for Gao's Lemma

- Recall the matrices

$$R_0 = \begin{bmatrix} s_0 & t_0 \\ s_1 & t_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Q_i = \begin{bmatrix} 0 & 1 \\ 1 & -q_i \end{bmatrix} \quad \text{for } i = 1, 2, \ldots, \ell$$

  and $R_i = Q_i Q_{i-1} \cdots Q_1 R_0 \in F[x]^{2\times 2}$ for $i = 0, 1, \ldots, \ell$ from the analysis of the traditional extended Euclidean algorithm in Problem Set 1

- We recall that for all $i = 0, 1, \ldots, \ell$ we have $R_i = \begin{bmatrix} s_i & t_i \\ s_{i+1} & t_{i+1} \end{bmatrix}$ and $R_i \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} r_i \\ r_{i+1} \end{bmatrix}$

- Since $\det Q_i = -1$ we have $\det R_i = (-1)^i$ and thus $R_i^{-1} = (-1)^i \begin{bmatrix} t_{i+1} & -t_i \\ -s_{i+1} & s_i \end{bmatrix}$ (exercise)

- We conclude that $s_{\ell+1} = (-1)^{\ell+1} r_1 / r_\ell$ and $t_{\ell+1} = (-1)^\ell r_0 / r_\ell$ (exercise)

Gao's Lemma

Lemma 13 (Gao [5])

Let $c, d, D \in \mathbb{Z}_{\geq 0}$ *and let* $q, r_0, r_1, f_0, f_1 \in F[x]$ *with* $\gcd(r_0, r_1) = 1$, $\deg q \geq D \geq c + d + 1$, *and* $\deg r_i \leq c$, $\deg f_i \leq d$ *for* $i = 0, 1$. *Run the extended Euclidean algorithm on input* $g_0 = qr_0 + f_0$ *and* $g_1 = qr_1 + f_1$ *to obtain the remainders* g_h *and* $g_{h+1} = s_{h+1}g_0 + t_{h+1}g_1$ *for* $s_{h+1}, t_{h+1} \in F[x]$ *with* $\deg g_h \geq D$ *and* $\deg g_{h+1} < D$. *Then,* $s_{h+1} = -\alpha r_1$ *and* $t_{h+1} = \alpha r_0$ *for some* $\alpha \in F \setminus \{0\}$

Proof.

Exercise □

Recap of Lecture 6

- Coping with **errors in data** using **error-correcting codes**
- A family of error-correcting codes (**Reed–Solomon codes**) based on evaluation–interpolation duality for univariate polynomials
  - Key observation: low-degree polynomials have few roots (exercise)
  - Fast **encoding** and **decoding** of Reed–Solomon codes via the fast univariate polynomial toolkit and **Gao's** (2003) **decoder**

Learning objectives (1/2)

- Terminology and objectives of modern algorithmics, including elements of algebraic, online, and randomised algorithms
- Ways of coping with uncertainty in computation, including error-correction and proofs of correctness
- The art of solving a large problem by reduction to one or more smaller instances of the same or a related problem
- (Linear) independence, dependence, and their abstractions as enablers of efficient algorithms

Learning objectives (2/2)

- Making use of duality
  - Often a problem has a corresponding **dual** problem that is obtainable from the original (the **primal**) problem by means of an easy transformation
  - The primal and dual control each other, enabling an algorithm designer to use the interplay between the two representations
- Relaxation and tradeoffs between objectives and resources as design tools
  - Instead of computing the exact optimum solution at considerable cost, often a less costly but principled approximation suffices
  - Instead of the complete dual, often only a randomly chosen partial dual or other relaxation suffices to arrive at a solution with high probability