Root finding over finite fields

Bruno Grenet
LIRMM
Université de Montpellier

Joris van der Hoeven & Grégoire Lecerf
CNRS – LIX
École polytechnique

GT ECo-Escape — February 11., 2015

Root finding over finite fields

Given $f \in \mathbb{F}_q[X]$, compute its roots, that is $\{\alpha \in \mathbb{F}_q : f(\alpha) = 0\}$.

Root finding over finite fields

Given $f \in \mathbb{F}_q[X]$, compute its roots, that is $\{\alpha \in \mathbb{F}_q : f(\alpha) = 0\}$.

- Building block for many algorithms in computer algebra: root finding over $\mathbb{Z}$, factorization, sparse interpolation, ...
- Applications in cryptography, error correcting codes, ...
- Derandomization
- Sparse interpolation: bottleneck in practice

[van der Hoeven & Lecerf, 2014]

$\mathbb{F}_q$: field with q elements, $q = p^r$ for some prime number p

- $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$; $+, -, \times$ and $/$ **modulo** p
- $\mathbb{F}_q \simeq \mathbb{F}_p[\lambda]/\langle\phi\rangle$ ($\phi \in \mathbb{F}_p[\lambda]$ irreducible of degree r); $+, -, \times$ and $/$ **modulo** p **and** ϕ

$\mathbb{F}_q$: field with q elements, $q = p^r$ for some prime number p

- $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$; $+, -, \times$ and / **modulo p**
- $\mathbb{F}_q \simeq \mathbb{F}_p[\lambda]/\langle\phi\rangle$ ($\phi \in \mathbb{F}_p[\lambda]$ irreducible of degree r); $+, -, \times$ and / **modulo p and ϕ**

- $\mathbb{F}_3 = \{0, 1, 2\}$:

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

and

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

;

$\mathbb{F}_q$: field with q elements, $q = p^r$ for some prime number p

- $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$; $+, -, \times$ and $/$ **modulo p**
- $\mathbb{F}_q \simeq \mathbb{F}_p[\lambda]/\langle\phi\rangle$ ($\phi \in \mathbb{F}_p[\lambda]$ irreducible of degree r); $+, -, \times$ and $/$ **modulo p and ϕ**

- $\mathbb{F}_3 = \{0, 1, 2\}$:

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

and

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

;

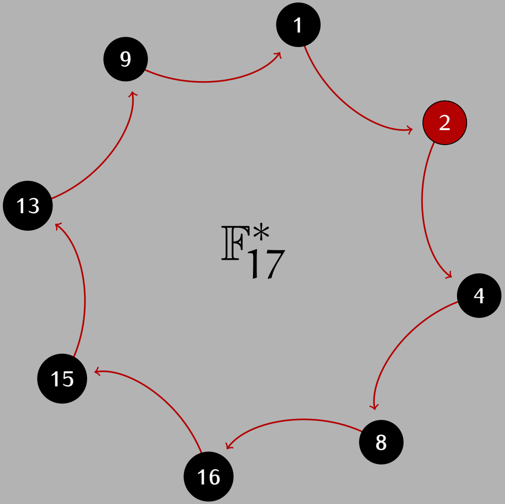
- $\mathbb{F}_4 = \mathbb{F}_2[\lambda]/\langle\lambda^2 + \lambda + 1\rangle = \{0, 1, \lambda, \lambda + 1\}$:

$+$	1	λ	$\lambda+1$
1	0	$\lambda+1$	λ
λ	$\lambda+1$	0	1
$\lambda+1$	λ	1	0

and

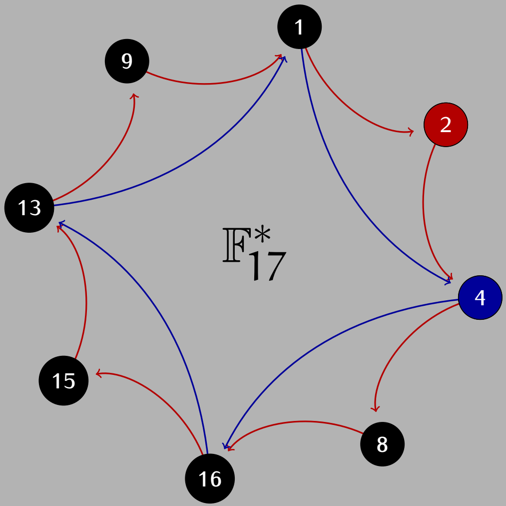
$\times$	λ	$\lambda+1$
λ	$\lambda+1$	1
$\lambda+1$	1	λ

.



2 is a primitive root of unity of order 8

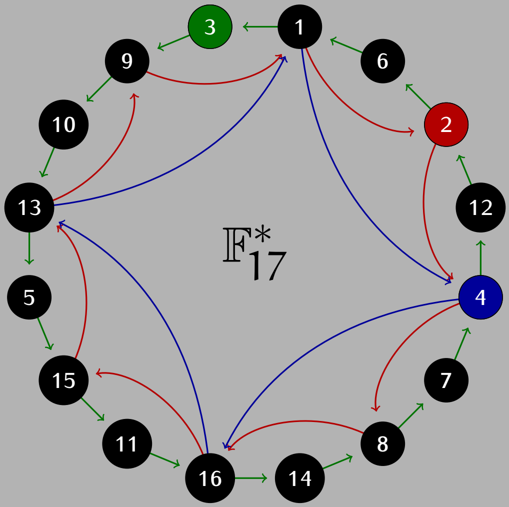
Multiplicative structure



2 is a primitive root of unity of order 8

4 is a primitive root of unity of order 4

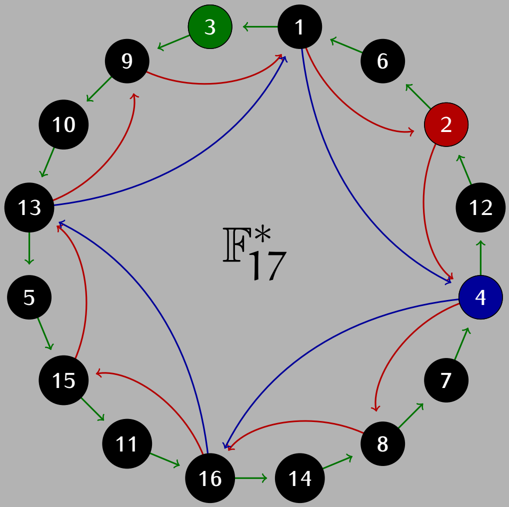
Multiplicative structure



2 is a primitive root of unity of order 8

4 is a primitive root of unity of order 4

3 is a primitive element of $\mathbb{F}_{17}^*$:
$\mathbb{F}_{17}^* = \{3^i : 0 \leqslant i < 16\}$



2 is a primitive root of unity of order 8

4 is a primitive root of unity of order 4

3 is a primitive element of $\mathbb{F}_{17}^*$: $\mathbb{F}_{17}^* = \{3^i : 0 \leqslant i < 16\}$

$$\alpha^{q-1} = \zeta^{i(q-1)} = 1$$

A first algorithm

Algorithm

Input: $f \in \mathbb{F}_q[X]$
Output: The roots of f.

$Z \leftarrow \emptyset$;
for all $\alpha \in \mathbb{F}_q^*$ do
  if $f(\alpha) = 0$ then
    Add α to Z;
return Z.

A first algorithm

Algorithm

Input: $f \in \mathbb{F}_q[X]$

Output: The roots of f.

```
Z ← ∅;
for all α ∈ F_q^* do
    if f(α) = 0 then
        Add α to Z;
return Z.
```

Theorem

The above algorithm runs in **deterministic time** $\mathrm{poly}(q, \deg(f))$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$

Output: The roots of f.

$Z \leftarrow \emptyset$;
for all $\alpha \in \mathbb{F}_q^*$ do
    if $f(\alpha) = 0$ then
        Add α to Z;
return Z.

Theorem

The above algorithm runs in **deterministic time** $\mathrm{poly}(q, \deg(f))$.

⚠ The input size is $(1 + \deg(f)) \log q$: exponential time!

Algorithm

Input: $f \in \mathbb{F}_q[X]$
Output: The roots of f.

1: $Z \leftarrow \emptyset$;
2: for all $\alpha \in \mathbb{F}_q^*$ do ▷ in **random** order
3: if $f(\alpha) = 0$ then
4: Add α to Z;
5: return Z.

Theorem

The above algorithm runs in **deterministic time** $\mathrm{poly}(q, \deg(f))$.

⚠ The input size is $(1 + \deg(f)) \log q$: exponential time!

The expected number of steps to discover all roots is $\frac{d}{d+1} q$.

Objectives

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.

Objectives

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.
- **Assumption:** $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i **distinct** and **nonzero**:
  - Easy reduction to this case: $f \leftarrow \gcd(f, X^{q-1} - 1)$ ($X^{q-1} - 1 = \prod_{\alpha \in \mathbb{F}_q^*} X - \alpha$).

Objectives

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.
- **Assumption:** $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i **distinct** and **nonzero**:
  - Easy reduction to this case: $f \leftarrow \gcd(f, X^{q-1} - 1)$ $(X^{q-1} - 1 = \prod_{\alpha \in \mathbb{F}_q^*} X - \alpha)$.
- **Extra input:** A primitive element ζ, or a primitive root of unity ξ of order χ.

Objectives

Obtain **fast** algorithms for polynomial root finding in finite fields.

- Deterministic, probabilistic, heuristic; in practice or in theory.
- **Assumption:** $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i **distinct** and **nonzero**:
  - Easy reduction to this case: $f \leftarrow \gcd(f, X^{q-1} - 1)$ $(X^{q-1} - 1 = \prod_{\alpha \in \mathbb{F}_q^*} X - \alpha)$.
- **Extra input:** A primitive element ζ, or a primitive root of unity ξ of order χ.
- **Smooth cardinality:**
  - $q = \rho\pi_1 \cdots \pi_m + 1$, where ρ, π_1, ..., π_m are *small*;
  - Practical purpose: $q = M2^m + 1$ is a *FFT prime*.

Algorithm

Input: $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i distinct and nonzero.

Algorithm

Input: $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i distinct and nonzero.

1: if $\deg(f) = 1$ then return its root;
2: $S \leftarrow \{s_1, \ldots, s_{(q-1)/2}\}$ taken at random in $\mathbb{F}_q^*$;
3: $u \leftarrow \prod_i (X - s_i)$ and $g \leftarrow \gcd(f, u)$;
4: return the union of the roots of g and f/g, recursively.

Algorithm

Input: $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i distinct and nonzero.

1: if $\deg(f) = 1$ then return its root;
2: $S \leftarrow \{s_1, \ldots, s_{(q-1)/2}\}$ taken at random in $\mathbb{F}_q^*$;
3: $u \leftarrow \prod_i (X - s_i)$ and $g \leftarrow \gcd(f, u)$;
4: return the union of the roots of g and f/g, recursively.

$$\mathbb{P}[\gcd(f,u) \in \{1, f\}] = \mathbb{P}[\forall i, \alpha_i \in S] + \mathbb{P}[\forall i, \alpha_i \notin S] = 1/2^{d-1}$$

Algorithm

Input: $f = \prod_{i=1}^{d}(X - \alpha_i)$, α_i distinct and nonzero.

1: if $\deg(f) = 1$ then return its root;
2: $S \leftarrow \{s_1, \ldots, s_{(q-1)/2}\}$ taken at random in $\mathbb{F}_q^*$;
3: $u \leftarrow \prod_i (X - s_i)$ and $g \leftarrow \gcd(f, u)$;
4: return the union of the roots of g and f/g, recursively.

$$\mathbb{P}[\gcd(f,u) \in \{1, f\}] = \mathbb{P}[\forall i, \alpha_i \in S] + \mathbb{P}[\forall i, \alpha_i \notin S] = 1/2^{d-1}$$

Good and bad news

The expected number of calls is $2d$.
The complexity of step 3 is $\tilde{O}(q)$.

Cantor-Zassenhaus' method

- $\prod_{\alpha \in \mathbb{F}_q^*} (X - \alpha) = X^{q-1} - 1$

Cantor-Zassenhaus' method

- $\prod_{\alpha\in\mathbb{F}_q^*}(X-\alpha) = X^{q-1}-1 = (X^{\frac{q-1}{2}}-1)(X^{\frac{q-1}{2}}+1)$ (q odd)

Cantor-Zassenhaus' method

- $\prod_{\alpha \in \mathbb{F}_q^*} (X - \alpha) = X^{q-1} - 1 = (X^{\frac{q-1}{2}} - 1)(X^{\frac{q-1}{2}} + 1)$ (q odd)
- With some luck, $\gcd(f, X^{\frac{q-1}{2}} - 1) \notin \{1, f\}$.

- $\prod_{\alpha \in \mathbb{F}_q^*} (X - \alpha) = X^{q-1} - 1 = (X^{\frac{q-1}{2}} - 1)(X^{\frac{q-1}{2}} + 1)$ (q odd)
- With some luck, $\gcd(f, X^{\frac{q-1}{2}} - 1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X + s)^{\frac{q-1}{2}} - 1)$ for some **random** $s \in \mathbb{F}_q$

- $\prod_{\alpha \in \mathbb{F}_q^*} (X - \alpha) = X^{q-1} - 1 = (X^{\frac{q-1}{2}} - 1)(X^{\frac{q-1}{2}} + 1)$ (q odd)
- With some luck, $\gcd(f, X^{\frac{q-1}{2}} - 1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X + s)^{\frac{q-1}{2}} - 1)$ for some **random** $s \in \mathbb{F}_q$

$$\#\left\{s \in \mathbb{F}_q : \gcd(f, (X + s)^{\frac{q-1}{2}} - 1) \notin \{1, f\}\right\} = \frac{q-1}{2}$$

- $\prod_{\alpha\in\mathbb{F}_q^*}(X-\alpha) = X^{q-1}-1 = (X^{\frac{q-1}{2}}-1)(X^{\frac{q-1}{2}}+1)$ (q odd)
- With some luck, $\gcd(f, X^{\frac{q-1}{2}}-1)\notin\{1,f\}$.
- Push your luck: $\gcd(f,(X+s)^{\frac{q-1}{2}}-1)$ for some **random** $s\in\mathbb{F}_q$

$$\#\left\{s\in\mathbb{F}_q : \gcd(f,(X+s)^{\frac{q-1}{2}}-1)\notin\{1,f\}\right\} = \frac{q-1}{2}$$

- $\gcd\in\{1,f\} \iff \begin{cases}\forall i, (\alpha_i+s)^{\frac{q-1}{2}} = 1 & \text{or}\\ \forall i, (\alpha_i+s)^{\frac{q-1}{2}} = -1;\end{cases}$
- $(\alpha_1+s)^{\frac{q-1}{2}} = -(\alpha_2+s)^{\frac{q-1}{2}} \neq 0 \iff \left(\frac{\alpha_1+s}{\alpha_2+s}\right)^{\frac{q-1}{2}} = -1;$
- $s\mapsto \frac{\alpha_1+s}{\alpha_2+s}$ is a bijection $\mathbb{F}_q\setminus\{-\alpha_2\}\to\mathbb{F}_q\setminus\{1\}$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Output: The roots of f.

1: if $\deg(f) = 1$ then return its root;
2: Take $s \in \mathbb{F}_q$ at random;
3: $h \leftarrow (X+s)^{\frac{q-1}{2}} \mod f$; ▷ repeated squaring
4: $g \leftarrow \gcd(f, h-1)$;
5: return the union of the roots of g and f/g.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Output: The roots of f.

1: if $\deg(f) = 1$ then return its root;
2: Take $s \in \mathbb{F}_q$ at random;
3: $h \leftarrow (X+s)^{\frac{q-1}{2}} \mod f$; ▷ repeated squaring
4: $g \leftarrow \gcd(f, h-1)$;
5: return the union of the roots of g and f/g.

Theorem

The above algorithm runs in expected time $\tilde{O}(d \log^2 q)$.

Modified Cantor-Zassenhaus' algorithm

Let $q = \chi\rho + 1$. Then $X^{q-1} - 1 = \prod_{i=0}^{\chi-1} (X^\rho - \xi^i)$, where $\xi^\chi = 1$.

Let $q = \chi\rho + 1$. Then $X^{q-1} - 1 = \prod_{i=0}^{\chi-1}(X^\rho - \xi^i)$, where $\xi^\chi = 1$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Extra input: A primitive root ξ of unity of order χ;
Output: The roots of f.

Modified Cantor-Zassenhaus' algorithm

Let $q = \chi\rho + 1$. Then $X^{q-1} - 1 = \prod_{i=0}^{\chi-1}(X^\rho - \xi^i)$, where $\xi^\chi = 1$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Extra input: A primitive root ξ of unity of order χ;
Output: The roots of f.

1: if $\deg(f) \leqslant 1$ then return its root;
2: Take $s \in \mathbb{F}_q$ at random;
3: $h \leftarrow (X+s)^\rho \mod f$; $g_0 \leftarrow f$;
4: for $i = 1$ to $\chi - 1$ do $g_i \leftarrow \gcd(g_0, h - \xi^i)$; $g_0 \leftarrow g_0/g_i$;
5: return the union of the roots of $g_0, \ldots, g_{\chi-1}$.

Let $q = \chi\rho + 1$. Then $X^{q-1} - 1 = \prod_{i=0}^{\chi-1}(X^\rho - \xi^i)$, where $\xi^\chi = 1$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Extra input: A primitive root ξ of unity of order χ;
Output: The roots of f.

1: if $\deg(f) \leqslant 1$ then return its root;
2: Take $s \in \mathbb{F}_q$ at random;
3: $h \leftarrow (X+s)^\rho \mod f$; $g_0 \leftarrow f$;
4: for $i = 1$ to $\chi - 1$ do $g_i \leftarrow \gcd(g_0, h - \xi^i)$; $g_0 \leftarrow g_0/g_i$;
5: return the union of the roots of $g_0, \dots, g_{\chi-1}$.

If $\chi \ll \log q/\log d$, the **speed-up is approximately** $\log_2 \chi$.

Definition

The **Graeffe transform** of $g \in \mathbb{F}_q[X]$ is the unique polynomial $h \in \mathbb{F}_q[X]$ such that

$$h(X^2) = g(X)g(-X).$$

If $g(X) = \prod_i (\alpha_i - X)$, then $h(X) = \prod_i (\alpha_i^2 - X)$.

Definition

The **Graeffe transform** of $g \in \mathbb{F}_q[X]$ is the unique polynomial $h \in \mathbb{F}_q[X]$ such that

$$h(X^2) = g(X)g(-X).$$

If $g(X) = \prod_i (\alpha_i - X)$, then $h(X) = \prod_i (\alpha_i^2 - X)$.

The **generalized Graeffe transform** of $g \in \mathbb{F}_q[X]$ of order π is

$$G_\pi(g)(X) = (-1)^{\pi \deg g} \operatorname{res}_z(g(z), z^\pi - x).$$

If $g = \prod_i (\alpha_i - X)$, then $G_\pi(g)(X) = \prod_i (\alpha_i^\pi - X)$.

The (generalized) Graeffe transform

Definition

The **Graeffe transform** of $g \in \mathbb{F}_q[X]$ is the unique polynomial $h \in \mathbb{F}_q[X]$ such that

$$h(X^2) = g(X)g(-X).$$

If $g(X) = \prod_i (\alpha_i - X)$, then $h(X) = \prod_i (\alpha_i^2 - X)$.

The **generalized Graeffe transform** of $g \in \mathbb{F}_q[X]$ of order π is

$$G_\pi(g)(X) = (-1)^{\pi \deg g} \operatorname{res}_z(g(z), z^\pi - x).$$

If $g = \prod_i (\alpha_i - X)$, then $G_\pi(g)(X) = \prod_i (\alpha_i^\pi - X)$.

Note. $G_{\pi_1 \pi_2} = G_{\pi_1} \circ G_{\pi_2}$

Let $q = \rho\pi_1 \cdots \pi_m + 1$.

Let $q = \rho\pi_1 \cdots \pi_m + 1$.

Remark

$G_{q-1}(g)(X) = \pm \prod_i (X - \alpha_i^{q-1}) = \pm(X-1)^{\deg(g)}$

Let $q = \rho\pi_1 \cdots \pi_m + 1$.

Remark

$G_{q-1}(g)(X) = \pm \prod_i (X - \alpha_i^{q-1}) = \pm (X-1)^{\deg(g)}$

Skeleton of the algorithms

1: $h_0 \leftarrow G_\rho(f)$;
2: for $i = 1$ to m do
3: $\quad h_i \leftarrow G_{\pi_i}(h_{i-1})$; $\triangleright\ h_i \leftarrow G_{\rho\pi_1\cdots\pi_i}(f)$
4: $Z_m \leftarrow \{1\}$, unique root of $h_m = G_{q-1}(f)$;
5: for $i = m-1$ down to 0 do
6: $\quad Z_i \leftarrow$ roots of h_i from Z_{i+1};
7: return the roots of f, computed from Z_0.

Lemma

Let π divide $q-1$, and ξ a primitive root of unity of order π. Then

$$G_\pi(g)(X^\pi) = g(X)g(\xi X)\cdots g(\xi^{\pi-1}X).$$

Lemma

Let π divide $q-1$, and ξ a primitive root of unity of order π. Then

$$G_\pi(g)(X^\pi) = g(X)g(\xi X)\cdots g(\xi^{\pi-1}X).$$

Theorem

Given $g \in \mathbb{F}_q[X]$ and a primitive root of unity ξ of order π, $G_\pi(g)$ can be computed in $\tilde{O}(\pi d \log q)$ operations.

Improved Graeffe transform computation

Theorem

Let $g \in \mathbb{F}_q[X]$ of degree d. For all $\delta > 0$ such that $d^{1+\delta} \leqslant q-1$, $G_\pi(g)$ can be computed in time $(d \log q)^{1+\delta} + \tilde{O}(d \log q \log \pi)$.

Improved Graeffe transform computation

Theorem

Let $g \in \mathbb{F}_q[X]$ of degree d. For all $\delta > 0$ such that $d^{1+\delta} \leqslant q-1$, $G_\pi(g)$ can be computed in time $(d \log q)^{1+\delta} + \tilde{O}(d \log q \log \pi)$.

Based on:

Theorem [Kedlaya-Umans'11]

Let f, g, $h \in \mathbb{F}_q[X]$ of degree d. For all $\delta > 0$, $(f \circ g \mod h)$ can be computed in time $d^{1+\delta}\tilde{O}(\log q)$.

Improved Graeffe transform computation

Theorem

Let $g \in \mathbb{F}_q[X]$ of degree d. For all $\delta > 0$ such that $d^{1+\delta} \leqslant q-1$, $G_\pi(g)$ can be computed in time $(d \log q)^{1+\delta} + \tilde{O}(d \log q \log \pi)$.

Based on:

Theorem [Kedlaya-Umans'11]

Let $f, g, h \in \mathbb{F}_q[X]$ of degree d. For all $\delta > 0$, $(f \circ g \mod h)$ can be computed in time $d^{1+\delta}\tilde{O}(\log q)$.

Corollary

Let $g \in \mathbb{F}_q[X]$ and $q = \rho\pi_1 \cdots \pi_m + 1$. For all δ, $G_\rho(g)$, $G_{\rho\pi_1}(g)$, …, $G_{\rho\pi_1\cdots\pi_{m-1}}(g)$ can be computed in time $(d \log^2 q)^{1+\delta}$.

Let $q = \rho\pi_1 \cdots \pi_m + 1 = \rho\chi + 1$ and $g = G_\rho(f) = \prod_i(\alpha_i - X)$

$$\prod_{i=1}^{r}(\alpha_i - X) \xrightarrow{G_\pi} \prod_{i=1}^{r}(\alpha_i^\pi - X)$$

Let $q = \rho\pi_1 \cdots \pi_m + 1 = \rho\chi + 1$ and $g = G_\rho(f) = \prod_i(\alpha_i - X)$

$$\prod_{i=1}^{r}(\alpha_i - X) \xrightarrow{G_\pi} \prod_{i=1}^{r}(\alpha_i^\pi - X)$$

ξ: primitive root of unity of order χ

$$\prod_{i=1}^{r}(\xi^{e_i} - X) \xrightarrow{G_\pi} \prod_{i=1}^{r}(\xi^{f_i} - X)$$

Let $q = \rho\pi_1 \cdots \pi_m + 1 = \rho\chi + 1$ and $g = G_\rho(f) = \prod_i(\alpha_i - X)$

$$\prod_{i=1}^{r}(\alpha_i - X) \xrightarrow{G_\pi} \prod_{i=1}^{r}(\alpha_i^\pi - X)$$

ξ: primitive root of unity of order χ

$$\prod_{i=1}^{r}(\xi^{e_i} - X) \xrightarrow{G_\pi} \prod_{i=1}^{r}(\xi^{f_i} - X)$$

$$\begin{aligned} & \forall i, (\xi^{e_i})^\pi = \xi^{f_i} \\ \Longleftrightarrow\quad & \forall i, \pi e_i = f_i \mod \chi \\ \Longleftrightarrow\quad & \forall i, e_i \in \left\{\frac{f_i + j\chi}{\pi} : 0 \leqslant j \leqslant \pi - 1\right\} \end{aligned}$$

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Extra input: A primitive root ξ of unity of order $\chi = \pi_1 \cdots \pi_m$;
Output: The ξ-logarithms of the roots of $G_\rho(f)$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Extra input: A primitive root ξ of unity of order $\chi = \pi_1 \cdots \pi_m$;
Output: The ξ-logarithms of the roots of $G_\rho(f)$.

1: $h_i \leftarrow$ Graeffe transform of f of order $\rho\pi_1 \cdots \pi_i$;
2: $E \leftarrow [0]$; ▷ ξ-log of the root of h_m
3: for $i = m$ down to 1 do
4: $\quad E \leftarrow \left[\frac{e + j\chi}{\pi_i} : e \in E, 0 \leqslant j \leqslant \pi_i - 1, h_{i-1}\left(\xi^{\frac{e+j\chi}{\pi_i}}\right) = 0\right]$;
5: return E.

A deterministic algorithm

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$;
Extra input: A primitive root ξ of unity of order $\chi = \pi_1 \cdots \pi_m$;
Output: The ξ-logarithms of the roots of $G_\rho(f)$.

1: $h_i \leftarrow$ Graeffe transform of f of order $\rho\pi_1 \cdots \pi_i$;
2: $E \leftarrow [0]$; ▷ ξ-log of the root of h_m
3: for $i = m$ down to 1 do
4: $E \leftarrow \left[\dfrac{e + j\chi}{\pi_i} : e \in E, 0 \leqslant j \leqslant \pi_i - 1, h_{i-1}\left(\xi^{\frac{e+j\chi}{\pi_i}}\right) = 0\right]$;
5: return E.

Theorem

If $\rho, \max_i \pi_i = O(\log q)$, the algorithm runs in time $\tilde{O}(d \log^3 q)$.

Lemma

Given $h = G_\pi(g)$, and $\{a_1, \dots, a_l\}$ its roots, one can compute the roots of g in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Lemma

Given $h = G_\pi(g)$, and $\{a_1, \dots, a_l\}$ its roots, one can compute the roots of g in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Theorem

Given $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$ and a primitive element of $\mathbb{F}_q^*$, the roots of f can be computed in time

$$\tilde{O}(\sqrt{S_1(q-1)} d \log^2 q) + (d \log^2 q)^{1+\delta}$$

where $S_1(q-1)$ is the largest factor of $q-1$.

Lemma

Given $h = G_\pi(g)$, and $\{a_1, \dots, a_l\}$ its roots, one can compute the roots of g in time $\tilde{O}(\sqrt{\pi} d \log q) + (d \log q)^{1+\delta}$ for all $\delta > 0$.

Theorem

Given $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$ and a primitive element of $\mathbb{F}_q^*$, the roots of f can be computed in time

$$\tilde{O}(\sqrt{S_1(q-1)} d \log^2 q) + (d \log^2 q)^{1+\delta}$$

where $S_1(q-1)$ is the largest factor of $q-1$.

- Best known bound for *smooth* q;
- If $q = M2^m - 1$, $M = O(\log q)$, complexity $\tilde{O}(d \log^2 q)$.

Definition

The **tangent Graeffe transform of order π** of $g \in \mathbb{F}_q[X]$ is

$$G_\pi(g(X+\varepsilon)) \in (\mathbb{F}_q[\varepsilon]/\langle\varepsilon^2\rangle)[X].$$

Remark. $G_\pi(g(X+\varepsilon)) = h(X) + \varepsilon\overline{h}(X)$ where $h = G_\pi(g)$.

Definition

The **tangent Graeffe transform of order π** of $g \in \mathbb{F}_q[X]$ is

$$G_\pi(g(X+\varepsilon)) \in (\mathbb{F}_q[\varepsilon]/\langle\varepsilon^2\rangle)[X].$$

Remark. $G_\pi(g(X+\varepsilon)) = h(X) + \varepsilon\overline{h}(X)$ where $h = G_\pi(g)$.

Lemma

A nonzero root β of h is a **simple root** iff $\overline{h}(\beta) \neq 0$. The corresponding root of g is $\alpha = \pi\beta h'(\beta)/\overline{h}(\beta)$.

Proof. $\overline{h}(\alpha^\pi) = \pi\alpha^{\pi-1}h'(\alpha^\pi)$.

- Replace f by $f_\tau(X) = f(X - \tau)$ for a random $\tau \in \mathbb{F}_q$.

- Replace f by $f_\tau(X) = f(X - \tau)$ for a random $\tau \in \mathbb{F}_q$.

Lemma

If $q = \rho\chi + 1$ with $\chi \geqslant d(d-1)$,

$$\mathbb{P}_{\tau \in \mathbb{F}_q}[G_\rho(f_\tau) \text{ has multiple roots}] \leqslant \frac{1}{2}.$$

- Replace f by $f_\tau(X) = f(X-\tau)$ for a random $\tau \in \mathbb{F}_q$.

Lemma

If $q = \rho\chi + 1$ with $\chi \geqslant d(d-1)$,

$$\mathbb{P}_{\tau\in\mathbb{F}_q}[G_\rho(f_\tau) \text{ has multiple roots}] \leqslant \frac{1}{2}.$$

Proof. Given $\alpha_i \neq \alpha_j$,

$$\#\left\{\tau \in \mathbb{F}_q : (\tau+\alpha_i)^\rho = (\tau+\alpha_j)^\rho\right\} \leqslant \rho.$$

$\implies$ $G_\rho(f_\tau)$ has multiple roots for at most $\frac{d(d-1)}{2}\rho$ values of τ.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$, $q = M2^m + 1$;
Extra input: ζ a primitive element of $\mathbb{F}_q^*$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$, $q = M2^m + 1$;
Extra input: ζ a primitive element of $\mathbb{F}_q^*$.

1: Find the smallest $\chi = M2^{m-l}$ s.t. $\chi \geqslant d(d-1)$;
2: $h_0 + \varepsilon \overline{h}_0 \leftarrow f(X - \tau + \varepsilon)$ for some random $\tau \in \mathbb{F}_q$;
3: Compute the Graeffe transforms $h_i + \varepsilon \overline{h}_i$ for $1 \leqslant i \leqslant m$;

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$, $q = M2^m + 1$;
Extra input: ζ a primitive element of $\mathbb{F}_q^*$.

1: Find the smallest $\chi = M2^{m-l}$ s.t. $\chi \geqslant d(d-1)$;
2: $h_0 + \varepsilon\overline{h}_0 \leftarrow f(X - \tau + \varepsilon)$ for some random $\tau \in \mathbb{F}_q$;
3: Compute the Graeffe transforms $h_i + \varepsilon\overline{h}_i$ for $1 \leqslant i \leqslant m$;

4: $E \leftarrow [e : h_m(\zeta^e) = 0]$; ▷ ζ-log of roots of h_m
5: $E \leftarrow \zeta$-log of roots of h_i, for $i = m-1$ to l;

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$, $q = M2^m + 1$;
Extra input: ζ a primitive element of $\mathbb{F}_q^*$.

1: Find the smallest $\chi = M2^{m-l}$ s.t. $\chi \geqslant d(d-1)$;
2: $h_0 + \varepsilon\overline{h}_0 \leftarrow f(X - \tau + \varepsilon)$ for some random $\tau \in \mathbb{F}_q$;
3: Compute the Graeffe transforms $h_i + \varepsilon\overline{h}_i$ for $1 \leqslant i \leqslant m$;

4: $E \leftarrow [e : h_m(\zeta^e) = 0]$; ▷ ζ-log of roots of h_m
5: $E \leftarrow \zeta$-log of roots of h_i, for $i = m-1$ to l;

6: $E \leftarrow \zeta$-log of **simple roots** of h_l; ▷ $\overline{h}_l(\zeta^e) \neq 0$
7: $Z \leftarrow$ corresponding roots of f; ▷ $\rho\zeta^e h_l'(\zeta^e)/\overline{h}_l(\zeta^e)$

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$, $q = M2^m + 1$;
Extra input: ζ a primitive element of $\mathbb{F}_q^*$.

1: Find the smallest $\chi = M2^{m-l}$ s.t. $\chi \geqslant d(d-1)$;
2: $h_0 + \varepsilon\overline{h}_0 \leftarrow f(X - \tau + \varepsilon)$ for some random $\tau \in \mathbb{F}_q$;
3: Compute the Graeffe transforms $h_i + \varepsilon\overline{h}_i$ for $1 \leqslant i \leqslant m$;

4: $E \leftarrow [e : h_m(\zeta^e) = 0]$; ▷ ζ-log of roots of h_m
5: $E \leftarrow \zeta$-log of roots of h_i, for $i = m-1$ to l;

6: $E \leftarrow \zeta$-log of **simple roots** of h_l; ▷ $\overline{h}_l(\zeta^e) \neq 0$
7: $Z \leftarrow$ corresponding roots of f; ▷ $\rho\zeta^e h_l'(\zeta^e)/\overline{h}_l(\zeta^e)$

8: Make a recursive call with $f / \prod_{a \in Z}(X - a)$.

Theorem

If $q = M2^m + 1$ with $M = O(\log q)$, the randomized algorithm runs in expected time $\tilde{O}(d \log^2 q)$.

Theorem

If $q = M2^m + 1$ with $M = O(\log q)$, the randomized algorithm runs in expected time $\tilde{O}(d \log^2 q)$.

- Same asymptotic as Cantor-Zassenhaus' algorithm;
- Better efficiency in practice.

Theorem

If $q = M2^m + 1$ with $M = O(\log q)$, the randomized algorithm runs in expected time $\tilde{O}(d \log^2 q)$.

- Same asymptotic as Cantor-Zassenhaus' algorithm;
- Better efficiency in practice.

Heuristic

Let $q = \rho\chi + 1$ and $f \in \mathbb{F}_q[X]$ with $d = \deg(f)$ roots in $\mathbb{F}_q^*$. If $\chi \geqslant 4d$, $G_\rho(f(X+\tau))$ has $\geqslant d/3$ simple roots with probability at least $1/2$, for a random $\tau \in \mathbb{F}_q$.

Justification: holds for a random f rather than $f(X+\tau)$.

Algorithm

Input: $f \in \mathbb{F}_q[X]$ with $\deg(f)$ distinct roots in $\mathbb{F}_q^*$, $q = M2^m + 1$;
Extra input: ζ a primitive element of $\mathbb{F}_q^*$.

1: Find the smallest $\chi = M2^{m-l}$ s.t. $\chi \geqslant 4d$;
2: $h_0 + \varepsilon\overline{h}_0 \leftarrow f(X - \tau + \varepsilon)$ for some random $\tau \in \mathbb{F}_q$;
3: Compute the Graeffe transform $h_l + \varepsilon\overline{h}_l$ of order 2^l;

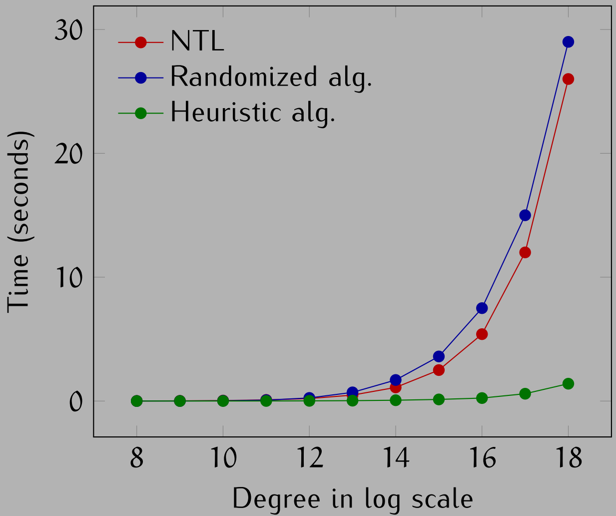
4: $E \leftarrow [e : h_l(\zeta^e) = 0]$; ▷ ζ-log of roots of h_l

5: $E \leftarrow \zeta$-log of **simple roots** of h_l; ▷ $\overline{h}_l(\zeta^e) \neq 0$
6: $Z \leftarrow$ corresponding roots of f; ▷ $\rho\zeta^e h_l'(\zeta^e)/\overline{h}_l(\zeta^e)$

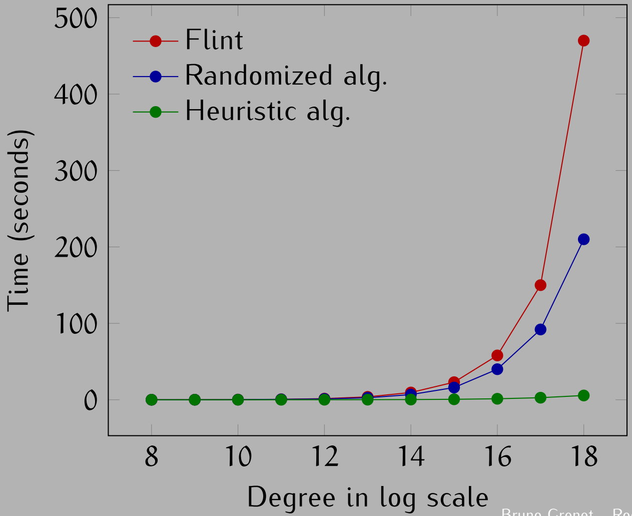
7: Make a recursive call with $f/\prod_{a \in Z}(X - a)$.

- Algorithms implemented in MATHEMAGIX (`http://mathemagix.org/`);
- Heuristic algorithm faster than FLINT and NTL by factors up to 80;
- Modification of Cantor-Zassenhaus algorithm: gain for large q only.

$q = 7 \cdot 2^{26} + 1$


NTL
Randomized alg.
Heuristic alg.
Time (seconds)
30
20
10
0
8
10
12
14
16
18
Degree in log scale

$q = 5 \cdot 2^{55} + 1$


Flint
Randomized alg.
Heuristic alg.
Time (seconds)
500
400
300
200
100
0
8
10
12
14
16
18
Degree in log scale

Conclusion

- Revisit classical algorithms for finite fields of smooth cardinality;

Conclusion

- Revisit classical algorithms for finite fields of smooth cardinality;
- New approach using Graeffe transforms:
  - Good deterministic complexity bounds;
  - Good probabilistic complexity bounds;
  - Good computation times.

- Revisit classical algorithms for finite fields of smooth cardinality;
- New approach using Graeffe transforms:
  - Good deterministic complexity bounds;
  - Good probabilistic complexity bounds;
  - Good computation times.
- Open questions:
  - Deterministic alg.: use of tangent Graeffe transforms;
  - Heuristic alg.: Graeffe transform of order 2^l is the bottleneck;
  - Prove the heuristic!

- Revisit classical algorithms for finite fields of smooth cardinality;
- New approach using Graeffe transforms:
  - Good deterministic complexity bounds;
  - Good probabilistic complexity bounds;
  - Good computation times.
- Open questions:
  - Deterministic alg.: use of tangent Graeffe transforms;
  - Heuristic alg.: Graeffe transform of order 2^l is the bottleneck;
  - Prove the heuristic!

Thank you!