Root finding over finite fields using Graeffe transforms

Bruno Grenet
LIRMM
Université de Montpellier

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

JNCF — Cluny — November 3., 2015

Statement of the problem

Root finding over finite fields

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

Statement of the problem

Root finding over finite fields

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

- **Assumption (A):** f is **monic**, **separable**, **splits** over $\mathbb{F}_q$, $f(0) \neq 0$:

$$f(X) = \prod_{i=1}^{d} (X - \alpha_i), \quad \alpha_i \in \mathbb{F}_q^*, \quad \alpha_i \neq \alpha_j$$

(easy reduction: $f \leftarrow \gcd(f, X^{q-1} - 1)$)

Statement of the problem

Root finding over finite fields

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

- **Assumption (A):** f is **monic**, **separable**, **splits** over $\mathbb{F}_q$, $f(0) \neq 0$:

$$f(X) = \prod_{i=1}^{d}(X - \alpha_i), \quad \alpha_i \in \mathbb{F}_q^*, \quad \alpha_i \neq \alpha_j$$

(easy reduction: $f \leftarrow \gcd(f, X^{q-1} - 1)$)

- Motivated by sparse interpolation **[van der Hoeven & Lecerf, 2014]**

State of the art & settings

- **No deterministic polytime algorithm is known** (even under ERH)

State of the art & settings

- **No deterministic polytime algorithm is known** (even under ERH)
- Randomized algorithms: $\tilde{O}(d\log^2 q)$ **in average** [**Rabin (1980)**]

State of the art & settings

- **No deterministic polytime algorithm is known** (even under ERH)
- Randomized algorithms: $\tilde{O}(d \log^2 q)$ **in average** [**Rabin (1980)**]
- Many factorization algorithms ⇝ no improvement for root finding [**Cantor-Zassenhaus (1981), Kaltofen-Shoup (1998), Kedlaya-Umans (2011)**]

State of the art & settings

- **No deterministic polytime algorithm is known** (even under ERH)
- Randomized algorithms: $\tilde{O}(d \log^2 q)$ **in average** [**Rabin (1980)**]
- Many factorization algorithms ⇝ no improvement for root finding
  [**Cantor-Zassenhaus (1981), Kaltofen-Shoup (1998), Kedlaya-Umans (2011)**]
- Better complexity bounds when $q - 1$ is sufficiently *smooth*
  [**Moenck (1977), von zur Gathen (1987), Mignotte-Schnorr (1988), Rónyai (1989), Shoup (1991, 1992), Źrałek (2010)**]

State of the art & settings

- **No deterministic polytime algorithm is known** (even under ERH)
- Randomized algorithms: $\tilde{O}(d \log^2 q)$ **in average** [**Rabin (1980)**]
- Many factorization algorithms ⇝ no improvement for root finding
  [**Cantor-Zassenhaus (1981), Kaltofen-Shoup (1998), Kedlaya-Umans (2011)**]
- Better complexity bounds when $q - 1$ is sufficiently *smooth*
  [**Moenck (1977), von zur Gathen (1987), Mignotte-Schnorr (1988), Rónyai (1989), Shoup (1991, 1992), Źrałek (2010)**]

- **FFT finite field**: $p = M \cdot 2^m + 1$ with $M = O(\log p)$
  - Useful in practice
  - Adapt old algorithms
  - New technique based on **Graeffe transforms**
  - Fast implementations

Rabin's algorithm

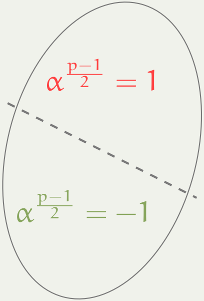
- $\prod_{\alpha \in \mathbb{F}_p^*} (X - \alpha) = X^{p-1} - 1$

Rabin's algorithm

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

Rabin's algorithm

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

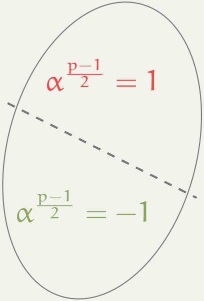


$\alpha^{\frac{p-1}{2}} = 1$

$\alpha^{\frac{p-1}{2}} = -1$

Rabin's algorithm

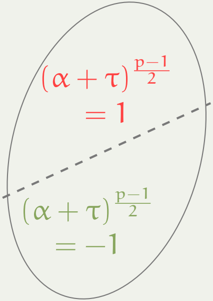
- $\prod_{\alpha\in\mathbb{F}_p^*}(X-\alpha) = X^{p-1}-1 = (X^{\frac{p-1}{2}}-1)(X^{\frac{p-1}{2}}+1)$



- With some luck, $\gcd(f, X^{\frac{p-1}{2}}-1)\notin\{1,f\}$.

Rabin's algorithm

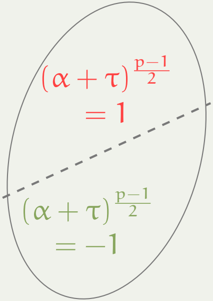
- $\prod_{\alpha\in\mathbb{F}_p^*}(X-\alpha) = X^{p-1}-1 = (X^{\frac{p-1}{2}}-1)(X^{\frac{p-1}{2}}+1)$


$(\alpha+\tau)^{\frac{p-1}{2}}$
$=1$
$(\alpha+\tau)^{\frac{p-1}{2}}$
$=-1$

- With some luck, $\gcd(f, X^{\frac{p-1}{2}}-1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X+\tau)^{\frac{p-1}{2}}-1)$ for some **random** $\tau\in\mathbb{F}_p$

Rabin's algorithm

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



$(\alpha+\tau)^{\frac{p-1}{2}} = 1$

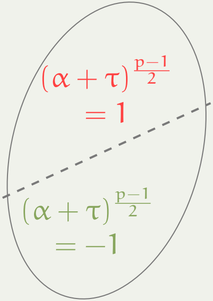
$(\alpha+\tau)^{\frac{p-1}{2}} = -1$

- With some luck, $\gcd(f, X^{\frac{p-1}{2}}-1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X+\tau)^{\frac{p-1}{2}}-1)$ for some **random** $\tau \in \mathbb{F}_p$

$$\deg\left(\gcd(f, (X+\tau)^{\frac{p-1}{2}}-1)\right) \simeq d/2$$

Rabin's algorithm

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



$(\alpha + \tau)^{\frac{p-1}{2}}$
$= 1$

$(\alpha + \tau)^{\frac{p-1}{2}}$
$= -1$

- With some luck, $\gcd(f, X^{\frac{p-1}{2}} - 1) \notin \{1, f\}$.
- Push your luck: $\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)$ for some **random** $\tau \in \mathbb{F}_p$

$$\deg\left(\gcd(f, (X + \tau)^{\frac{p-1}{2}} - 1)\right) \simeq d/2$$

Randomized algorithm

The roots of $f \in \mathbb{F}_p[X]$ can be computed in expected time $\tilde{O}(d \log^2 p)$.

Modified Rabin's algorithm

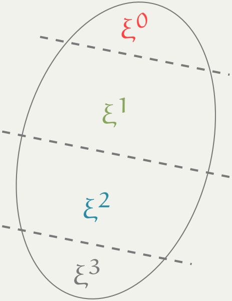
(for FFT finite fields)

$$X^{p-1}-1=\prod_{i=0}^{2^\ell-1}(X^{M2^{m-\ell}}-\xi^i),\text{ where } \xi \text{ is primitive of order } 2^\ell.$$

Modified Rabin's algorithm

(for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i), \text{ where } \xi \text{ is primitive of order } 2^\ell.$$



$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^0)$

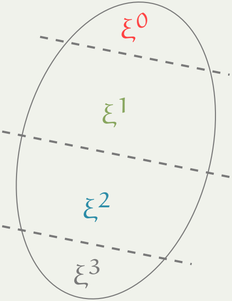
$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^1)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^2)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^3)$

Modified Rabin's algorithm
(for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i),$$ where ξ is primitive of order 2^ℓ.



$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^0)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^1)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^2)$

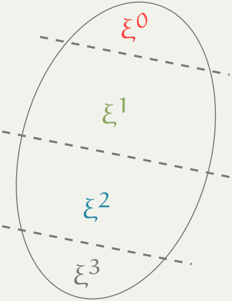
$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^3)$

degrees
$\simeq d/2^\ell$

Modified Rabin's algorithm

(for FFT finite fields)

$$X^{p-1} - 1 = \prod_{i=0}^{2^\ell - 1} (X^{M2^{m-\ell}} - \xi^i)\text{, where } \xi \text{ is primitive of order } 2^\ell.$$



$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^0)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^1)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^2)$

$\gcd(f, (X+\tau)^{M2^{m-\ell}} - \xi^3)$

degrees
$\simeq d/2^\ell$

Worthwhile in practice for **small** $\ell = 2, 3, \ldots$

The Graeffe transform

Let $f(X) = \prod_i (X - \alpha_i) \in \mathbb{F}_p[X]$.

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

The Graeffe transform

Let $f(X) = \prod_i (X - \alpha_i) \in \mathbb{F}_p[X]$.

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Definition

$G_2(f)(X) = \prod_i (X - \alpha_i^2)$ is the **Graeffe transform** of f.

The Graeffe transform

Let $f(X) = \prod_i (X - \alpha_i) \in \mathbb{F}_p[X]$.

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Definition

$G_2(f)(X) = \prod_i (X - \alpha_i^2)$ is the **Graeffe transform** of f.

$G_\rho(f)(X) = \prod_i (X - \alpha_i^\rho)$ is the **Graeffe transform of order** ρ of f.

The Graeffe transform

Let $f(X) = \prod_i (X - \alpha_i) \in \mathbb{F}_p[X]$.

$$f(X)f(-X) = \prod_i (X - \alpha_i)(-X - \alpha_i) = (-1)^d \prod_i (X^2 - \alpha_i^2)$$

Definition

$G_2(f)(X) = \prod_i (X - \alpha_i^2)$ is the **Graeffe transform** of f.

$G_\rho(f)(X) = \prod_i (X - \alpha_i^\rho)$ is the **Graeffe transform of order** ρ of f.

Remarks:

- $G_{\rho_1\rho_2} = G_{\rho_1} \circ G_{\rho_2}$, and in particular $G_{2^\ell} = G_2 \circ \cdots \circ G_2$
- $G_{p-1}(f)(X) = \prod_i (X - \alpha_i^{p-1}) = (X - 1)^d$

Using Graeffe transforms

$$f \xrightarrow{G_2} g_1 \xrightarrow{G_2} g_2 \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_m \xrightarrow{G_M} g_{m+1}$$

Using Graeffe transforms

$$\begin{tikzcd}
f \arrow[r, "G_2"] & g_1 \arrow[r, "G_2"] & g_2 \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m \arrow[r, "G_M"] & g_{m+1} \arrow[d, mapsto] \\
 & & & & & \{1\}
\end{tikzcd}$$

Using Graeffe transforms

$$\begin{tikzcd}
f \arrow[r, "G_2"] & g_1 \arrow[r, "G_2"] & g_2 \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m \arrow[r, "G_M"] & g_{m+1} \arrow[d, mapsto] \\
Z(f) & Z_1 \arrow[l] & Z_2 \arrow[l] & \cdots \arrow[l] & Z_m \arrow[l] & \{1\} \arrow[l]
\end{tikzcd}$$

Using Graeffe transforms

$$\begin{tikzcd}
f \arrow[r, "G_2"] & g_1 \arrow[r, "G_2"] & g_2 \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m \arrow[r, "G_M"] & g_{m+1} \arrow[d, mapsto] \\
Z(f) & Z_1 \arrow[l] & Z_2 \arrow[l] & \cdots \arrow[l] & Z_m \arrow[l] & \{1\} \arrow[l]
\end{tikzcd}$$

- $Z_m \subseteq \{\zeta^{i2^m} : 0 \leqslant i \leqslant M-1\}$ where ζ is a primitive element of $\mathbb{F}_p^*$

Using Graeffe transforms

$$\begin{tikzcd}
f \arrow[r, "G_2"] & g_1 \arrow[r, "G_2"] & g_2 \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m \arrow[r, "G_M"] & g_{m+1} \arrow[d, mapsto] \\
Z(f) & Z_1 \arrow[l] & Z_2 \arrow[l] & \cdots \arrow[l] & Z_m \arrow[l] & \{1\} \arrow[l]
\end{tikzcd}$$

- $Z_m \subseteq \{\zeta^{i2^m} : 0 \leqslant i \leqslant M-1\}$ where ζ is a primitive element of $\mathbb{F}_p^*$
- For $\beta \in Z_{k+1}$,
  - $\gcd(g_k, X^2 - \beta) = \begin{cases} X - \alpha_i & \text{(simple root)} \\ (X-\alpha_i)(X-\alpha_j) & \text{(multiple root)} \end{cases}$

Using Graeffe transforms

$$\begin{tikzcd}
f \arrow[r, "G_2"] & g_1 \arrow[r, "G_2"] & g_2 \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m \arrow[r, "G_M"] & g_{m+1} \arrow[d, mapsto] \\
Z(f) & Z_1 \arrow[l] & Z_2 \arrow[l] & \cdots \arrow[l] & Z_m \arrow[l] & \{1\} \arrow[l]
\end{tikzcd}$$

- $Z_m \subseteq \{\zeta^{i2^m} : 0 \leqslant i \leqslant M-1\}$ where ζ is a primitive element of $\mathbb{F}_p^*$
- For $\beta \in Z_{k+1}$,
  - $\gcd(g_k, X^2 - \beta) = \begin{cases} X - \alpha_i & \text{(simple root)} \\ (X-\alpha_i)(X-\alpha_j) & \text{(multiple root)} \end{cases}$
  - If $\beta = \zeta^e$, $\alpha_i, \alpha_j \in \{\zeta^{e/2}, \zeta^{(e+2^m M)/2}\}$

Deterministic complexity

Improvements and generalization:

- Modular composition for Graeffe transforms **[Kedlaya-Umans (2008)]**
- Fast discrete logarithms in $\mathbb{F}_q^*$ **[Pohlig-Hellman (1978)]**
- Computation of roots *à la* Pollard-Strassen **[Shoup (1991)]**

Deterministic complexity

Improvements and generalization:

- Modular composition for Graeffe transforms [Kedlaya-Umans (2008)]
- Fast discrete logarithms in $\mathbb{F}_q^*$ [Pohlig-Hellman (1978)]
- Computation of roots *à la* Pollard-Strassen [Shoup (1991)]

Theorem

Given $f \in \mathbb{F}_q[X]$ satisfying $(\mathcal{A})$, the irreducible factorization of $(q-1)$ and a primitive element of $\mathbb{F}_q^*$, the roots of f can be computed in time

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

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

Deterministic complexity

Improvements and generalization:

- Modular composition for Graeffe transforms [Kedlaya-Umans (2008)]
- Fast discrete logarithms in $\mathbb{F}_q^*$ [Pohlig-Hellman (1978)]
- Computation of roots *à la* Pollard-Strassen [Shoup (1991)]

Theorem

Given $f \in \mathbb{F}_q[X]$ satisfying $(\mathcal{A})$, the irreducible factorization of $(q-1)$ and a primitive element of $\mathbb{F}_q^*$, the roots of f can be computed in time

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

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

- Refines Shoup's complexity bounds
- Note: If $q = M \cdot 2^m + 1$, $M = O(\log q)$, complexity $\tilde{O}(d\log^2 q)$.

Tangent Graeffe transform

Definition

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

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

Tangent Graeffe transform

Definition

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

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

Remarks:

- $(f + \varepsilon f')(X) = f(X + \varepsilon)$
- $G_2(f + \varepsilon f') = G_2(f) + \varepsilon g$ with $g(X^2) = f(X)f'(-X) + f(-X)f'(X)$

Tangent Graeffe transform

Definition

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

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

Remarks:

- $(f + \varepsilon f')(X) = f(X + \varepsilon)$
- $G_2(f + \varepsilon f') = G_2(f) + \varepsilon g$ with $g(X^2) = f(X)f'(-X) + f(-X)f'(X)$

Lemma

Let $g + \varepsilon \overline{g} = G_{2^\ell}(f + \varepsilon f')$. A nonzero root β of g is **simple iff** $\overline{g}(\beta) \neq 0$. The corresponding root of f is $\alpha = 2^\ell \beta g'(\beta)/\overline{g}(\beta)$.

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Lemma

If $2^\ell \leqslant \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geqslant 1/2$.

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Lemma

If $2^\ell \leqslant \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geqslant 1/2$.

$$f(X+\tau+\varepsilon) \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_\ell + \varepsilon \overline{g}_\ell \xrightarrow{G_2} \cdots \xrightarrow{G_2} g_m + \varepsilon \overline{g}_m$$

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Lemma

If $2^\ell \leqslant \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geqslant 1/2$.

$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_\ell+\varepsilon\overline{g}_\ell \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m+\varepsilon\overline{g}_m \arrow[d, mapsto] \\
 & & & & Z_m \\
 & & & & \{\xi^e : 0 \leqslant e < M\} \arrow[u, phantom, "\cap"]
\end{tikzcd}$$

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Lemma

If $2^\ell \leqslant \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geqslant 1/2$.

$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_\ell + \varepsilon\overline{g}_\ell \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m + \varepsilon\overline{g}_m \arrow[d, mapsto] \\
 & & Z_\ell & \cdots \arrow[l] & Z_m \arrow[l] \arrow[d, phantom, "\cap"] \\
 & & & & \{\xi^e : 0 \leqslant e < M\}
\end{tikzcd}$$

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Lemma

If $2^\ell \leqslant \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geqslant 1/2$.

$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_\ell + \varepsilon\overline{g}_\ell \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m + \varepsilon\overline{g}_m \arrow[d, mapsto] \\
Z_0 & & Z_\ell \arrow[ll, "\text{Only simple roots}"'] & \cdots \arrow[l] & Z_m \arrow[l] \arrow[d, phantom, "\cap"] \\
 & & & & \{\xi^e : 0 \leqslant e < M\}
\end{tikzcd}$$

Randomized algorithm

Goal: Ensure many **simple roots**.

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

Lemma

If $2^\ell \leqslant \frac{p-1}{d(d-1)}$, $G_{2^\ell}(f_\tau)$ has **no multiple root** with prob. $\geqslant 1/2$.

$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_\ell + \varepsilon\overline{g}_\ell \arrow[r, "G_2"] & \cdots \arrow[r, "G_2"] & g_m + \varepsilon\overline{g}_m \arrow[d, mapsto] \\
Z_0 \arrow[u, bend left, "\text{recursive call: } f/\prod_{\alpha\in Z_0}(X-\alpha)"'] & & Z_\ell \arrow[ll, "\text{Only simple roots}"] & \cdots \arrow[l] & Z_m \arrow[l] \arrow[d, phantom, "\cap"] \\
& & & & \{\xi^e : 0 \leqslant e < M\}
\end{tikzcd}$$

Randomized complexity

Theorem

Given $f \in \mathbb{F}_p[X]$ satisfying (A) and a primitive element of $\mathbb{F}_p^*$, the randomized algorithm runs in **expected time** $\tilde{O}(d \log^2 p)$, for $p = M \cdot 2^m + 1$ with $M = O(\log p)$.

Randomized complexity

Theorem

Given $f \in \mathbb{F}_p[X]$ satisfying (A) and a primitive element of $\mathbb{F}_p^*$, the randomized algorithm runs in **expected time** $\tilde{O}(d \log^2 p)$, for $p = M \cdot 2^m + 1$ with $M = O(\log p)$.

- Same asymptotic as Rabin's algorithm
- Better efficiency in practice
- Primitive elements easy to compute in practice

Heuristic algorithm

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geqslant 1/2$, for a random $\tau \in \mathbb{F}_p$.

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

Heuristic algorithm

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geqslant 1/2$, for a random $\tau \in \mathbb{F}_p$.

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

$$f(X+\tau+\varepsilon) \xrightarrow{\quad G_{2^\ell} \quad} g_\ell + \varepsilon \overline{g}_\ell$$

Heuristic algorithm

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geqslant 1/2$, for a random $\tau \in \mathbb{F}_p$.

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

$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[r, "G_{2^\ell}"] & g_\ell + \varepsilon\overline{g}_\ell \arrow[d, mapsto] \\
 & Z_\ell \\
 & \cap \\
 & \{\xi^e : 0 \leqslant e < M \cdot 2^\ell\}
\end{tikzcd}$$

Heuristic algorithm

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geqslant 1/2$, for a random $\tau \in \mathbb{F}_p$.

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

$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[rr, "G_{2^\ell}"] & & g_\ell + \varepsilon\overline{g}_\ell \arrow[d, mapsto] \\
Z_0 & & Z_\ell \arrow[ll, "\text{only simple roots}"'] \arrow[d, phantom, "\cap"] \\
 & & \{\xi^e : 0 \leqslant e < M \cdot 2^\ell\}
\end{tikzcd}$$

Heuristic algorithm

Heuristic

If $2^\ell \simeq p/d$, $G_{2^\ell}(f(X+\tau))$ has $\Omega(d)$ **simple roots** with probability $\geqslant 1/2$, for a random $\tau \in \mathbb{F}_p$.

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

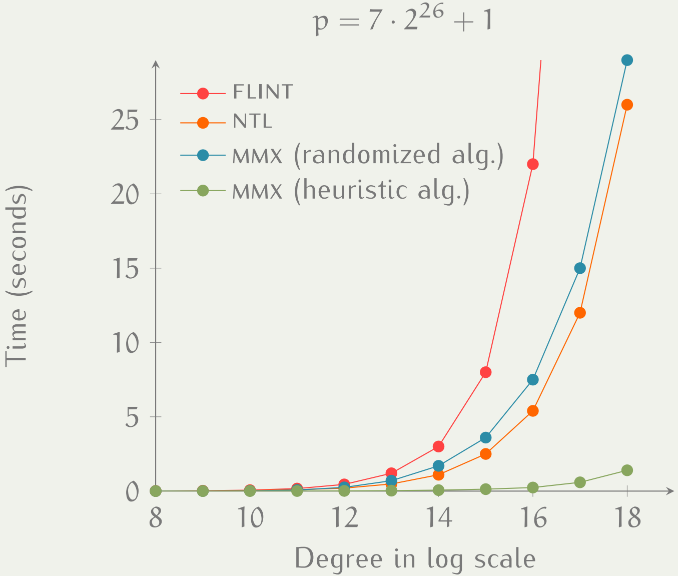
$$\begin{tikzcd}
f(X+\tau+\varepsilon) \arrow[r, "G_{2^\ell}"] & g_\ell + \varepsilon\overline{g}_\ell \arrow[d, mapsto] \\
Z_0 \arrow[u, bend left, "\begin{array}{c}\text{recursive call:}\\ f/\prod_{\alpha\in Z_0}(X-\alpha)\end{array}"] & Z_\ell \arrow[l, "\text{only simple roots}"'] \arrow[d, phantom, "\cap"] \\
 & \{\xi^e : 0 \leqslant e < M\cdot 2^\ell\}
\end{tikzcd}$$

Heuristic complexity

Theorem

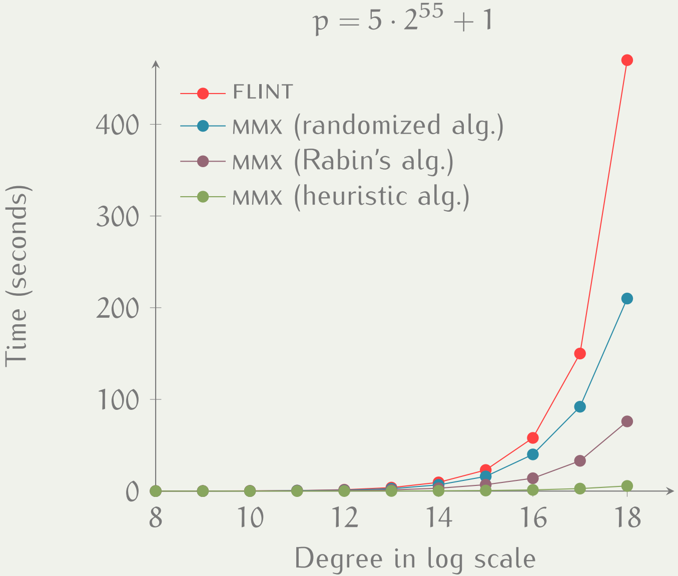
Suppose that f is chosen at random in $\mathbb{F}_p[X]$ or that the heuristic holds. Given a primitive element of $\mathbb{F}_p^*$, the heuristic algorithm runs in **expected time** $\tilde{O}(d \log^2 p)$, for $p = M \cdot 2^m + 1$ with $M = O(\log p)$.

Timings



$p = 7 \cdot 2^{26} + 1$
FLINT
NTL
MMX (randomized alg.)
MMX (heuristic alg.)
Time (seconds)
0
5
10
15
20
25
8
10
12
14
16
18
Degree in log scale

Timings


$p = 5 \cdot 2^{55} + 1$
FLINT
MMX (randomized alg.)
MMX (Rabin's alg.)
MMX (heuristic alg.)
Time (seconds)
400
300
200
100
0
8
10
12
14
16
18
Degree in log scale

Conclusion

- Revisit classical algorithms for FFT finite fields

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - Good deterministic complexity bounds
  - Good probabilistic complexity bounds
  - Good running times

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - Good deterministic complexity bounds
  - Good probabilistic complexity bounds
  - Good running times
- Source code in C++, in MATHEMAGIX

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - Good deterministic complexity bounds
  - Good probabilistic complexity bounds
  - Good running times
- Source code in C++, in MATHEMAGIX
- Root finding is not the bottleneck for sparse interpolation anymore

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - Good deterministic complexity bounds
  - Good probabilistic complexity bounds
  - Good running times
- Source code in C++, in Mathemagix
- Root finding is not the bottleneck for sparse interpolation anymore
- Open questions:
  - Deterministic alg.: use of tangent Graeffe transforms
  - Heuristic alg.: Graeffe transform of order 2^l is the bottleneck
  - Prove the heuristic

Conclusion

- Revisit classical algorithms for FFT finite fields
- New approach using Graeffe transforms
  - Good deterministic complexity bounds
  - Good probabilistic complexity bounds
  - Good running times
- Source code in C++, in Mathemagix
- Root finding is not the bottleneck for sparse interpolation anymore
- Open questions:
  - Deterministic alg.: use of tangent Graeffe transforms
  - Heuristic alg.: Graeffe transform of order 2^l is the bottleneck
  - Prove the heuristic

Merci de votre attention !