Génération d'éléments normaux en C

21 février 2017

Table des matières

Introduction
- Contexte
- Éléments normaux

Génération d'éléments normaux
- Algorithme randomisé
- Algorithmes déterministes

Résultats

CONTEXTE

On travail dans des *extensions de corps finis*

$$\mathbb{F}_{p^d}/\mathbb{F}_p$$

où $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$ est le corps à p éléments, et plus généralement $\mathbb{F}_{p^d} \cong \mathbb{F}_p[X]/(P(X))$ (où P est un polynôme irréductible de degré d de $\mathbb{F}_p[X]$) est le corps à p^d éléments.

RAPPELS ET NOTATIONS

Proposition

Le corps $\mathbb{F}_{p^d}$ *est un* $\mathbb{F}_p$*-espace vectoriel de dimension* d *: une base est* $\{1, X, \ldots, X^{d-1}\}$.

Définition (automorphisme de Frobenius)

On note σ *l'automorphisme de Frobenius*

$$\begin{array}{rccc} \sigma : & \mathbb{F}_{p^d} & \to & \mathbb{F}_{p^d} \\ & x & \mapsto & x^p. \end{array}$$

Éléments normaux : définition

Définition (élément normal)

Soit $\alpha \in \mathbb{F}_{p^d}$, on dit que α est un *élément normal* si $\{\alpha, \sigma(\alpha), \ldots, \sigma^{d-1}(\alpha)\} = \left\{\alpha, \alpha^p, \ldots, \alpha^{p^{d-1}}\right\}$ est une base de $\mathbb{F}_{p^d}$.

Ce type de base est appelé *base normale*.

ÉLÉMENTS NORMAUX : À QUOI ÇA SERT ?

- *En pratique*, les bases normales rendent l'arithmétique (additions, multiplications, etc) moins coûteuses en énergie. Applications en cryptographie, théorie des codes.
- *En théorie*, elles sont utiles pour comprendre des problèmes d'algèbre liés aux extensions de corps.

Existence d'éléments normaux

Théorème

Soit $\mathbb{F}_{p^d}/\mathbb{F}_p$ une extension de corps finie. Il existe un élément normal α pour cette extension.

Le but du projet était donc de réussir à générer de tels éléments.

PRÉSENTATION DES ALGORITHMES

On va voir trois algorithmes.

- Algorithme randomisé $O((d + \log p)(d \log p)^2)$
- Algorithme de Lüneburg $O((d^2 + \log p)(d \log p)^2)$
- Algorithme de Lenstra $O((d^2 + \log p)(d \log p)^2)$

Présentation des algorithmes

On va voir trois algorithmes.

- Algorithme randomisé $O((d + \log p)(d \log p)^2)$
- Algorithme de Lüneburg $O((d^2 + \log p)(d \log p)^2)$
- Algorithme de Lenstra $O((d^2 + \log p)(d \log p)^2)$

UNE PROPOSTION UTILE

Proposition

Soit $\alpha \in \mathbb{F}_{p^d}$ et $P = \alpha^{p^{d-1}} X^{d-1} + \cdots + \alpha^p X + \alpha$. L'élément α est normal si et seulement si P et $X^d - 1$ sont premiers entre eux.

On en déduit un algorithme de test pour savoir si un élément est normal.

ALGORITHME RANDOMISÉ NAÏF

On déduit également un algorithme pour générer des éléments normaux.

ALGORITHME RANDOMISÉ NAÏF

On déduit également un algorithme pour générer des éléments normaux.

- Prendre un élément $\alpha \in \mathbb{F}_{p^d}$ au hasard.

ALGORITHME RANDOMISÉ NAÏF

On déduit également un algorithme pour générer des éléments normaux.

- Prendre un élément $\alpha \in \mathbb{F}_{p^d}$ au hasard.
- Vérifier s'il est normal, en utilisant la proposition précédente.

ALGORITHME RANDOMISÉ NAÏF

On déduit également un algorithme pour générer des éléments normaux.

- Prendre un élément $\alpha \in \mathbb{F}_{p^d}$ au hasard.
- Vérifier s'il est normal, en utilisant la proposition précédente.
- S'il est normal, on le garde ; sinon, on recommence.

Algorithme randomisé naïf

On déduit également un algorithme pour générer des éléments normaux.

- Prendre un élément $\alpha \in \mathbb{F}_{p^d}$ au hasard.
- Vérifier s'il est normal, en utilisant la proposition précédente.
- S'il est normal, on le garde ; sinon, on recommence.

Mais on peut faire mieux.

ALGORITHME RANDOMISÉ ÉLABORÉ

Proposition

Soit $f(X)$ un polynôme irréductible de degré d de $\mathbb{F}_p[X]$ et α une racine de f. Soit

$$g(X) = \frac{f(X)}{(X-\alpha)f'(\alpha)}.$$

Il y a alors au moins $p - d(d-1)$ éléments $u \in \mathbb{F}_p$ tels que $g(u)$ soit un élément normal de $\mathbb{F}_{p^d}$.

Ainsi, si $p > 2d(d-1)$, on a au moins une chance sur deux que $g(u)$ soit un élément normal.

Algorithme randomisé élaboré

On a ainsi un algorithme plus efficace.

Algorithme randomisé élaboré

On a ainsi un algorithme plus efficace.

- Prendre un élément $u \in \mathbb{F}_p$ au hasard.

ALGORITHME RANDOMISÉ ÉLABORÉ

On a ainsi un algorithme plus efficace.

- Prendre un élément $u \in \mathbb{F}_p$ au hasard.
- Vérifier si $g(u)$ est normal.

ALGORITHME RANDOMISÉ ÉLABORÉ

On a ainsi un algorithme plus efficace.

- Prendre un élément $u \in \mathbb{F}_p$ au hasard.
- Vérifier si $g(u)$ est normal.
- Si $g(u)$ est normal, on s'arrête ; sinon, on recommence.

ALGORITHME RANDOMISÉ ÉLABORÉ

On a ainsi un algorithme plus efficace.

- Prendre un élément $u \in \mathbb{F}_p$ au hasard.
- Vérifier si $g(u)$ est normal.
- Si $g(u)$ est normal, on s'arrête ; sinon, on recommence.

Mais pour garantir que cet algorithme fonctionne, il faut au moins que $p > d(d-1)$.

POLYNÔMES DE σ-ORDRE

Définition (polynôme de σ-ordre)

Soit $\theta \in \mathbb{F}_{p^d}$ un élément. Soit k le plus petit entier positif tel que $\sigma^k(\theta) = \theta^{p^k}$ appartienne au sous-espace vectoriel engendré par $\{\sigma^i(\theta) \mid 0 \leq i < k\}$. Si $\sigma^k(\theta) = \sum_{i=0}^{k-1} c_i \sigma^i(\theta)$, alors on définit le polynôme de σ-ordre de θ comme

$$\mathrm{Ord}_\theta(X) = X^k - \sum_{i=0}^{k-1} c_i X^i.$$

POLYNÔMES DE σ-ORDRE : PROPRIÉTÉS

Proposition

On a les propriétés suivantes :

- *L'élément θ est normal si et seulement si* $\mathrm{Ord}_\theta = X^d - 1$.
- *Si* $P(\sigma)\theta = 0$, *alors* Ord_θ *divise* P.
- *Soit* $\alpha, \beta \in \mathbb{F}_{p^d}$, $\mathrm{Ord}_{\alpha+\beta}$ *divise* $\mathrm{ppcm}(\mathrm{Ord}_\alpha, \mathrm{Ord}_\beta)$.
- *Si* Ord_α *et* Ord_β *sont premiers entre eux,* $\mathrm{Ord}_{\alpha+\beta} = \mathrm{Ord}_\alpha \mathrm{Ord}_\beta$.

Algorithme de Lüneburg

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.

Algorithme de Lüneburg

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \ldots, f_{d-1}) = X^d - 1$.

ALGORITHME DE LÜNEBURG

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \ldots, f_{d-1}) = X^d - 1$.
- Écrire, pour chaque i

$$f_i = \prod_{j=1}^{r} g_j^{e_{i,j}}$$

où les g_j sont premiers entre eux deux à deux.

Algorithme de Lüneburg

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \dots, f_{d-1}) = X^d - 1$.
- Écrire, pour chaque i

$$f_i = \prod_{j=1}^{r} g_j^{e_{i,j}}$$

  où les g_j sont premiers entre eux deux à deux.
- Trouver $i(j)$ tel que $e_{i(j),j}$ soit maximal parmi les $e_{i,j}$.

Algorithme de Lüneburg

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \ldots, f_{d-1}) = X^d - 1$.
- Écrire, pour chaque i

$$f_i = \prod_{j=1}^{r} g_j^{e_{i,j}}$$

  où les g_j sont premiers entre eux deux à deux.
- Trouver $i(j)$ tel que $e_{i(j),j}$ soit maximal parmi les $e_{i,j}$.
- Calculer $h_j = f_{i(j)} / g_j^{e_{i(j),j}}$ puis $\beta_j = h_j(\sigma) X^{i(j)}$

Algorithme de Lüneburg

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \ldots, f_{d-1}) = X^d - 1$.
- Écrire, pour chaque i

$$f_i = \prod_{j=1}^{r} g_j^{e_{i,j}}$$

  où les g_j sont premiers entre eux deux à deux.
- Trouver $i(j)$ tel que $e_{i(j),j}$ soit maximal parmi les $e_{i,j}$.
- Calculer $h_j = f_{i(j)} / g_j^{e_{i(j),j}}$ puis $\beta_j = h_j(\sigma) X^{i(j)}$
  - On a $\mathrm{Ord}_{\beta_j} = g_j^{e_{i(j),j}}$.

ALGORITHME DE LÜNEBURG

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \ldots, f_{d-1}) = X^d - 1$.
- Écrire, pour chaque i

$$f_i = \prod_{j=1}^{r} g_j^{e_{i,j}}$$

  où les g_j sont premiers entre eux deux à deux.
- Trouver $i(j)$ tel que $e_{i(j),j}$ soit maximal parmi les $e_{i,j}$.
- Calculer $h_j = f_{i(j)}/g_j^{e_{i(j),j}}$ puis $\beta_j = h_j(\sigma)X^{i(j)}$
  - On a $\mathrm{Ord}_{\beta_j} = g_j^{e_{i(j),j}}$.
- Alors $\beta = \sum_{j=1}^{r} \beta_j$ est normal.

ALGORITHME DE LÜNEBURG

- Générer $f_i = \mathrm{Ord}_{X^i}$, pour $0 \leq i < d$.
  - On a $\mathrm{ppcm}(f_0, \ldots, f_{d-1}) = X^d - 1$.
- Écrire, pour chaque i

$$f_i = \prod_{j=1}^{r} g_j^{e_{i,j}}$$

  où les g_j sont premiers entre eux deux à deux.
- Trouver $i(j)$ tel que $e_{i(j),j}$ soit maximal parmi les $e_{i,j}$.
- Calculer $h_j = f_{i(j)}/g_j^{e_{i(j),j}}$ puis $\beta_j = h_j(\sigma)X^{i(j)}$
  - On a $\mathrm{Ord}_{\beta_j} = g_j^{e_{i(j),j}}$.
- Alors $\beta = \sum_{j=1}^{r} \beta_j$ est normal.
  - En effet $\mathrm{Ord}_\beta = \prod_{j=1}^{r} \mathrm{Ord}_{b_j} = \prod_{j=1}^{r} g_j^{e_{i(j),j}} = X^d - 1$.

ALGORITHME DE LENSTRA

L'algorithme de Lenstra fonctionne de la façon suivante.

- On prend un élément $\theta \in \mathbb{F}_{p^d}$ quelconque et on calcule Ord_θ.

ALGORITHME DE LENSTRA

L'algorithme de Lenstra fonctionne de la façon suivante.

- On prend un élément $\theta \in \mathbb{F}_{p^d}$ quelconque et on calcule Ord_θ.
- Si $\mathrm{Ord}_\theta = X^d - 1$ on s'arrête ; sinon, on modifie θ pour que Ord_θ monte de degré.

ALGORITHME DE LENSTRA

L'algorithme de Lenstra fonctionne de la façon suivante.

- On prend un élément $\theta \in \mathbb{F}_{p^d}$ quelconque et on calcule Ord_θ.
- Si $\mathrm{Ord}_\theta = X^d - 1$ on s'arrête ; sinon, on modifie θ pour que Ord_θ monte de degré.

Comme Ord_θ divise $X^d - 1$, l'algorithme s'arrête et on a bien un élément normal à la fin. Les modifications qu'on fait à θ sont liées à de l'algèbre linéaire : on doit résoudre des systèmes linéaires.

Résultats

- Les complexités en pratique et théoriques semblent en adéquation.
- De manière surprenante l'algorithme randomisé naïf va aussi vite que l'algorithme élaboré, après étude : les algorithmes naïf et élaboré semblent trouver des éléments normaux du premier coup.
- L'algorithme de Lenstra est plus rapide et plus stable que celui de Lüneburg.
- L'algorithme de Lüneburg est un peu chaotique : temps de calculs parfois étonnamment longs, ou courts.

The End