

# Recherche d'un point de torsion dans une courbe définie par un polynôme lacunaire

Louis Leroux

## 1 Introduction et résultats principaux

On cherche ici à déterminer si un polynôme à deux variables s'annule en un point de torsion, c'est à dire un point dont les deux coordonnées sont des racines de l'unité. Pour  $m \in \mathbb{N}^*$ , on notera dans la suite :

$$\mu_m = \{\zeta \in \mathbb{C} : z^m = 1\}$$

et on utilisera également la notation :

$$\mu_\infty = \bigcup_{m \in \mathbb{N}^*} \mu_m.$$

On se donne dans la suite un polynôme  $F(x, y) = \sum_{i=1}^N a_i x^{\alpha_i} y^{\beta_i} \in \mathbb{Z}[x, y]$  de degré  $d$ , possédant  $N$  termes non nuls et de hauteur  $h(F)$ , où la hauteur de  $F$  est définie par :

$$h(F) = \max_{1 \leq i \leq N} \{\log |a_i|\}.$$

L'algorithme que nous allons décrire a, pour  $N$  fixé, une complexité polynômiale en  $\log d$  et linéaire en  $h(f)$ . On utilise dans la suite la notation  $O_N(w(d, N, h))$  pour désigner une fonction majorée par  $w(d, N, h)$  fois une fonction de  $N$ .

**Théorème 1.1** *Il existe un algorithme qui a la propriété suivante : étant donné un polynôme  $F(x, y) \in \mathbb{Z}[x, y]$  de degré  $d$ , de hauteur  $h$  et possédant au plus  $N$  termes non nuls, l'algorithme détermine si  $F$  s'annule en un point de torsion en au plus*

$$O_N \left( h(\log d)^{\delta \sqrt{\frac{N}{\log N}}} \right)$$

opérations binaires, où  $\delta$  est un réel strictement positif effectivement calculable.

De plus, dans le cas où  $F$  s'annule en un point de torsion, l'algorithme renvoie  $(m, a, b) \in \mathbb{N}^* \times \mathbb{N}^2$  tel que :

$$F(\zeta_m^a, \zeta_m^b) = 0.$$

où  $\zeta_m$  est une racine de l'unité d'ordre  $m$ .

Cet algorithme sera décrit dans la section 3.3.

Ce problème est étudié dans [BS02] et dans cet article, les auteurs donnent une borne pour le nombre de points de torsion contenus dans une courbe et décrivent un algorithme pour déterminer ces points. Leur méthode nécessite des calculs de résultants de polynômes qui peuvent être lourds si ces polynômes ont un degré élevé. Ainsi la complexité de leur algorithme, qui n'est pas explicitée dans l'article, semble être polynômiale en  $d$ .

Pour démontrer le théorème 1.1, on va distinguer deux cas selon que le polynôme s'annule en un nombre fini de points de torsion ou non. Dans tout ce qui suit,  $n \in \mathbb{N}^*$  et pour  $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}^n$ , on note  $\underline{x}^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$  et  $\underline{x} = (x_1, \dots, x_n)$ . Si  $f(\underline{x}) \in \mathbb{Z}[x_1^{\pm 1}, \dots, x_n^{\pm 1}]$ , on utilise la notation introduite par Schinzel en posant  $Jf(\underline{x}) = \underline{x}^\alpha f(\underline{x})$  où  $\alpha \in \mathbb{Z}^n$  est tel que  $\underline{x}^\alpha f(\underline{x}) \in \mathbb{Z}[\underline{x}]$  et n'est pas divisible par un monôme. Les *polynômes cyclotomiques généralisés* sont alors les polynômes de la forme  $J\Phi_m(\underline{x}^\beta)$  où  $m \in \mathbb{N}^*$ ,  $\Phi_m$  désigne le  $m$ -ième polynôme cyclotomique et  $\beta \in \mathbb{Z}^n$  est primitif. On peut montrer que ces polynômes sont irréductibles. Un problème posé par Lang consistait à montrer que si une courbe plane  $F(x, y) = 0$  possède une infinité de points de torsion, alors le polynôme  $F$  possède un facteur cyclotomique généralisé. Ce problème a été résolu positivement par Ihara, Serre, Tate, Laurent... (voir [La83]) On va donc distinguer les deux cas suivants :

- *Le polynôme  $F$  possède un facteur cyclotomique généralisé.* Dans ce cas, on déterminera un facteur cyclotomique généralisé de  $F$  puis on trouvera un point de torsion qui annule ce facteur.
- *Le polynôme  $F$  ne possède pas de facteur cyclotomique généralisé.* Dans ce cas, on utilisera une majoration pour l'ordre des éventuels points de torsion en lesquels s'annule  $F$  et on en déterminera un le cas échéant.

Pour traiter le premier cas, on aura donc besoin de savoir déterminer les facteurs cyclotomiques généralisés d'un polynôme lacunaires à plusieurs variables. Dans [FS04], Filaseta et Schinzel décrivent un algorithme, récemment

amélioré avec Granville dans [FGS08], qui permet de déterminer les facteurs cyclotomiques d'un polynôme lacunaire d'une seule variable. On va le généraliser pour un polynôme lacunaire à plusieurs variables.

**Théorème 1.2** *Il existe un algorithme qui a la propriété suivante : étant donné un polynôme  $F(x_1, \dots, x_n) \in \mathbb{Z}[x_1, \dots, x_n]$  de degré  $d$ , de hauteur  $h$  et possédant au plus  $N$  termes non nuls, l'algorithme détermine si le polynôme  $F$  possède un facteur cyclotomique généralisé en au plus*

$$O_N(n \log d (\log \log d)^2 \log \log \log d + h)$$

*opérations binaires.*

*De plus, dans le cas où  $F(\underline{x})$  possède un facteur cyclotomique généralisé, l'algorithme renvoie l'ensemble :*

$$S_F = \{(\beta, S_\beta) : \beta \in \mathbb{Z}^n \text{ et } \forall m \in S_\beta, J\Phi_m(\underline{x}^\beta) | F(\underline{x})\}.$$

On a utilisé dans ce théorème la notation  $O_N(w(d, h))$  pour désigner une fonction majorée par  $w(d, h)$  fois une fonction de  $N$ .

Introduisons maintenant les notions de *support*, de *polytope de Newton* et de *volume*. Soit  $F(x, y) \in \mathbb{Z}[x, y]$  de degré  $d$  tel que

$$F(x, y) = \sum_{i=1}^N a_i x^{\alpha_i} y^{\beta_i} \quad \text{où } a_i \neq 0, \forall 1 \leq i \leq N.$$

On appelle *support* de  $F$  et on note

$$\text{Supp}(F) = \{(\alpha_i, \beta_i) : 1 \leq i \leq N\} \subset \mathbb{Z}^2.$$

On définit alors le *polytope de Newton* du polynôme  $F$ , noté  $N(F)$ , comme l'enveloppe convexe du support de  $F$ . On considère également  $\text{Vol}(N(F))$ , le volume du polytope de Newton de  $F$  et on remarque que  $\text{Vol}(N(F)) \leq \frac{1}{2}d^2$ . On peut noter au passage que ces notions se généralisent en dimension supérieure.

Beukers et Smyth montrent dans [BS02] qu'un polynôme à deux variables ne possédant pas de facteur cyclotomique généralisé s'annule en au plus  $22\text{Vol}(N(F))$  points de torsion. De plus, si un polynôme à coefficients entiers s'annule en un point de torsion, il s'annule également sur l'orbite de

ce point sous l'action du groupe de Galois  $\text{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$ . On déduit de cela que si  $F(x, y) \in \mathbb{Z}[x, y]$ , de degré  $d$  ne possédant pas de facteur cyclotomique généralisé, s'annule en un point de torsion  $\zeta \in \mu_\infty^2$ . Alors on a

$$[\mathbb{Q}(\zeta) : \mathbb{Q}] \leq 22\text{Vol}(F) \leq 11d^2.$$

Comme l'ensemble  $Z_d = \{\zeta \in \mu_\infty^2 : [\mathbb{Q}(\zeta) : \mathbb{Q}] \leq 11d^2\}$  est fini, cette proposition permet de déterminer si un polynôme à deux variables s'annule en un point de torsion de la façon suivante : pour chaque élément  $\zeta$  de  $Z_d$ , on teste la nullité de  $F(\zeta)$ . La complexité de cet algorithme est alors exponentielle en  $\log d$  puisque le cardinal de  $Z_d$  est exponentiel en  $\log d$ . On va déterminer une condition supplémentaire satisfaite par certains des points de torsion en lesquels s'annule le polynôme  $F$  pour accélérer cet algorithme.

## 2 Facteurs cyclotomiques généralisés

Le but de cette partie est de démontrer le théorème 1.2. On va se ramener à l'algorithme en une variable de Filaseta, Granville et Schinzel en utilisant un lemme dû à Kalfoten et Koiran. Les algorithmes que nous allons décrire nécessitent des multiplications d'entiers et des calculs de pgcd. Nous utiliserons pour cela les deux résultats classiques suivants :

- Le produit de deux entiers de taille  $s$  se calcule en

$$O(s \log s \log \log s)$$

opérations binaires à l'aide de l'algorithme de Schönhage-Strassen [SS71].

- Le pgcd de deux entiers de taille  $s$  se calcule en

$$O(s(\log s)^2 \log \log s)$$

opérations binaires à l'aide de l'algorithme de Knuth-Schönhage [K70].

### 2.1 Résultats préliminaires

On se donne un polynôme  $F(\underline{x}) \in \mathbb{Z}[\underline{x}]$  de degré  $d$ , et de hauteur  $h$  et possédant au plus  $N$  termes non nuls. On cherche alors à déterminer, s'ils existent,  $m \in \mathbb{N}$  et  $\beta \in \mathbb{Z}^n$  tels que :

$$J\Phi_m(\underline{x}^\beta) | F(\underline{x}).$$

Pour cela, on commence par déterminer les valeurs possibles de  $\beta$  grâce au lemme 4 de [KK06] dont nous redonnons la démonstration.

**Lemme 2.1** Soient  $F(\underline{x}) = \sum_{i=1}^N a_i \underline{x}^{\alpha_i} \in \mathbb{Z}[\underline{x}]$ ,  $m \in \mathbb{N}^*$  et  $\beta \in \mathbb{Z}^n$  un vecteur primitif tels que  $J\Phi_m(\underline{x}^\beta) | F(\underline{x})$ . Alors si  $\beta_n \neq 0$ , il existe un entier  $j$  vérifiant  $1 \leq j \leq N$ ,  $\alpha_{1,n} \neq \alpha_{j,n}$  et  $\forall 1 \leq i \leq n$ ,

$$\beta_i = \frac{\alpha_{1,i} - \alpha_{j,i}}{k} \quad \text{où } k = \text{pgcd}_{1 \leq i \leq n}(\alpha_{1,i} - \alpha_{j,i}).$$

**Démonstration :** Soient  $F(\underline{x}) = \sum_{i=1}^N a_i \underline{x}^{\alpha_i} \in \mathbb{Z}[\underline{x}]$ ,  $m \in \mathbb{N}^*$  et  $\beta \in \mathbb{Z}^n$  à coordonnées primitives tels que  $J\Phi_m(\underline{x}^\beta) | F(\underline{x})$ . Dans  $\overline{\mathbb{Q}}[\underline{x}]$ , le polynôme  $J\Phi_m(\underline{x}^\beta)$  se factorise de la façon suivante en produit d'irréductibles :

$$J\Phi_m(\underline{x}^\beta) = \prod_{\zeta \in \mu_m} J(\underline{x}^\beta - \zeta).$$

Soient  $\zeta \in \mu_m$  et  $\lambda \in \overline{\mathbb{Q}}$  tels que  $\lambda^{\beta_n} = \zeta$ . Comme  $J(\underline{x}^\beta - \zeta)$  est un facteur de  $F(\underline{x})$  dans  $\overline{\mathbb{Q}}[\underline{x}]$ , on obtient

$$x_n = \lambda x_1^{-\beta_1/\beta_n} \dots x_{n-1}^{-\beta_{n-1}/\beta_n}$$

comme racine de  $F$  dans une clôture algébrique de  $\overline{\mathbb{Q}}[x_1, \dots, x_{n-1}]$ . En développant en cette racine le premier terme  $\underline{x}^{\alpha_1}$  de  $F$  dans le corps des séries de Puiseux en  $x_1, \dots, x_{n-1}$ , on obtient

$$\alpha_{1,i} - \frac{\beta_i}{\beta_n} \alpha_{1,n}$$

comme exposant fractionnaire de  $x_i$ , pour  $1 \leq i \leq n-1$ . Puisque  $x_n$  est une racine de  $F$ , ce terme doit être annulé par un autre dont on appelle l'indice  $j$ , où  $1 \leq j \leq N$ . Comme tous les exposants de ces deux monômes doivent être identiques, on a

$$\forall 1 \leq i \leq n-1, \quad \alpha_{1,i} - \alpha_{j,i} = \frac{\beta_i}{\beta_n} (\alpha_{1,n} - \alpha_{j,n}).$$

De plus,  $\alpha_{1,n} \neq \alpha_{j,n}$  car  $\alpha_{1,i} \neq \alpha_{j,i}$  pour un certain  $i$  puisque sinon  $\alpha_1 = \alpha_j$ . On a donc  $\beta_i = 0$  si et seulement si  $\alpha_{1,i} = \alpha_{j,i}$ . Posons  $k = \frac{(\alpha_{1,n} - \alpha_{j,n})}{\beta_n}$ . On a

$$\forall 1 \leq i \leq N, \quad k\beta_i = \alpha_{1,i} - \alpha_{j,i}.$$

Cette dernière égalité implique que  $k$  est entier car sinon son dénominateur serait un diviseur commun à tous les  $\beta_i$  ce qui serait contraire à l'hypothèse faite sur  $\beta$ . Ainsi  $k$  divise tous les  $\alpha_{1,i} - \alpha_{j,i}$  et comme les  $\beta_i$  sont premiers entre eux,  $k$  est bien le pgcd de ces nombres.  $\square$

La preuve de ce lemme donne clairement un algorithme pour déterminer les exposants des éventuels facteurs cyclotomiques généralisés de  $F$ . De plus, les facteurs pour lesquels  $\beta_n = 0$  sont aussi couverts par ce lemme puisque dans ce cas, si  $j$  est donné, il existe  $1 \leq i \leq N$  tel que  $\alpha_{1,i} \neq \alpha_{j,i}$  et  $x_i$  peut alors jouer le rôle de  $x_n$ . En tout, on obtient donc au plus  $N - 1$  vecteurs possibles. Tout ceci est résumé dans le lemme suivant :

**Lemme 2.2** *Il existe un algorithme qui à la propriété suivante : étant donné un polynôme à  $n$  variables  $F(\underline{x}) = \sum_{i=1}^N a_i \underline{x}^{\alpha_i} \in \mathbb{Z}[\underline{x}]$  de degré  $d > 1$ , l'algorithme détermine les vecteurs  $\beta \in \mathbb{Z}^n$  du lemme 2.1 en*

$$O(nN \log d (\log \log d)^2 \log \log \log d)$$

*opérations binaires.*

L'algorithme est le suivant :

**Entrée :** Un polynôme  $F(\underline{x}) = \sum_{i=1}^N a_i \underline{x}^{\alpha_i} \in \mathbb{Z}[\underline{x}]$  donné par la liste de ses coefficients non nuls, affectés des exposants correspondants.

**Sortie :** Les vecteurs  $\beta \in \mathbb{Z}^n$  du lemme 2.1.

1. Faire  $V \leftarrow \emptyset$ .
2. **Pour**  $2 \leq j \leq N$  faire :
  - (a)  $k \leftarrow \text{pgcd}_{1 \leq i \leq n}(\alpha_{1,i} - \alpha_{j,i})$ .
  - (b) **Pour**  $1 \leq i \leq n$ , faire  $\beta_i \leftarrow \frac{\alpha_{1,i} - \alpha_{j,i}}{k}$ .

$$(c) V \leftarrow V \cup \{\beta\}.$$

3. Renvoyer  $V$  et terminer.

**Démonstration :** émontrons que cet algorithme vérifie bien les propriétés du lemme précédent. On effectue dans cet algorithme les mêmes calculs que dans la preuve du lemme 2.1. On détermine donc bien l'ensemble de vecteurs voulu.

Montrons qu'on ne dépasse pas la complexité annoncée. Une fois fixé  $j$  à l'étape 2, le calcul des différences  $\alpha_{1,i} - \alpha_{j,i}$  nécessite  $O(n \log d)$  opérations binaires car ces nombres sont majorés par  $d$ . Pour calculer  $k$ , on effectue récursivement  $n - 1$  calculs de pgcd de couples d'entiers majorés par  $d$ . Ceci se fait donc en  $O(n \log d (\log \log d)^2 \log \log \log d)$  opérations binaires. Le calcul des  $\beta_i$  se fait ensuite en  $O(n \log d \log \log d \log \log \log d)$  opérations binaires car on effectue  $n$  additions et  $n$  divisions sur des entiers majorés par  $d$ . De plus, l'étape 2 est répétée  $N - 1$  fois. On effectue donc au total,  $O(nN \log d (\log \log d)^2 \log \log \log d)$  opérations binaires.  $\square$

Comme le remarquent Kaltofen et Koiran dans [KK06], en faisant la même substitution que dans la démonstration du lemme 2.1 et en exprimant le fait que la série de Puiseux qui en résulte est identiquement nulle, on obtient :

$$\sum_{i=1}^N a_i \lambda^{\alpha_{i,n}} x_1^{\alpha_{i,1} - \alpha_{i,n} \beta_1 / \beta_n} \dots x_{n-1}^{\alpha_{i,n-1} - \alpha_{i,n} \beta_{n-1} / \beta_n} = 0.$$

Après avoir regroupé les monômes de même degré, on obtient un système d'au plus  $N - 1$  polynômes en l'indéterminée  $\lambda$  et ayant chacun au plus  $N$  termes non nuls. Dans cet article, ils montrent également le résultat suivant :

**Lemme 2.3** *Le polynôme  $J\Phi_m(\underline{x}^\beta)$  est un facteur de  $F(\underline{x})$  si et seulement si il existe un nombre complexe  $\lambda$  solution du système précédent où  $\lambda^{\beta_n}$  est une racine de l'unité d'ordre  $m$ .*

On est ainsi ramené à déterminer les facteurs cyclotomiques de polynômes lacunaires à une seule variable, ce que l'on peut faire grâce au résultat de Filaseta, Granville et Schinzel dont nous rappellerons l'énoncé après avoir introduit la notation suivante : pour  $u$  et  $v$  entiers positifs, on note

$$D(u, v) = \{m \in \mathbb{N} : u|m \text{ et } m|v\}.$$

On remarque que  $D(u, v) = \emptyset$  si et seulement si  $u \nmid v$  et que si  $u, v, u', v' \in \mathbb{N}$ , on a

$$D(u, v) \cap D(u', v') = D(\text{ppcm}(u, u'), \text{pgcd}(v, v')).$$

Ce résultat se généralise par récurrence de la façon suivante : si  $u_1, \dots, u_n, v_1, \dots, v_n \in \mathbb{N}$  alors :

$$\bigcap_{i=1}^n D(u_i, v_i) = D(\text{ppcm}(u_i), \text{pgcd}(v_i)).$$

Rappelons maintenant l'énoncé du théorème C de [FGS08].

**Théorème 2.4** *Il existe un algorithme qui a la propriété suivante : étant donné un polynôme  $F(x) \in \mathbb{Z}[x]$  de degré  $d$ , de hauteur  $h$  et possédant au plus  $N$  termes non nuls, l'algorithme détermine si ce polynôme possède un facteur cyclotomique en au plus*

$$O_N(\log d(\log \log d)^2 \log \log \log d + h)$$

*opérations binaires.*

*De plus, dans le cas où  $F$  possède un facteur cyclotomique, l'algorithme renvoie l'ensemble :*

$$S_F = \{(u, v) \in \mathbb{N}^2 : \forall m \in D(u, v), \Phi_m(x) | F(x)\}.$$

On va maintenant décrire l'algorithme du théorème 1.2 puis prouver sa correction et enfin évaluer sa complexité.

## 2.2 Algorithme

**Entrée :** Un polynôme  $F(\underline{x}) = \sum_{i=1}^N a_i \underline{x}^{\alpha_i} \in \mathbb{Z}[\underline{x}]$  donné par la liste de ses coefficients non nuls, affectés des exposants correspondants.

**Sortie :** L'ensemble  $\{(\beta, S_\beta) : \beta \in \mathbb{Z}^n \text{ et } \forall m \in S_\beta, J\Phi_m(\underline{x}^\beta) | F(\underline{x})\}$ .

1. Faire  $S_F \leftarrow \emptyset$ .
2. Déterminer l'ensemble  $V$  des candidats possibles pour  $\beta$  à l'aide de l'algorithme 2.2.

3. Choisir  $\beta \in V$ , faire  $V \leftarrow V \setminus \{\beta\}$  et former le système d'équations à une variable comme dans le lemme 2.1.
4. **Pour** chaque polynôme  $f$  du système obtenu, faire

$$S_f \leftarrow \{(u, v) \in \mathbb{N}^2 : \forall m \in D(u, v), \Phi_m | f\}.$$

grâce à l'algorithme 2.4.

5. Déterminer l'ensemble  $S_\beta$ , intersection de tous les ensembles  $S_f$  obtenus à l'étape précédente et faire  $S_F \leftarrow S_F \cup \{(\beta, S_\beta)\}$ .
6. **Si**  $V = \emptyset$ , **renvoyer**  $S_F$  et terminer, **sinon retourner** à l'étape 3.

### 2.3 Correction

Montrons que l'algorithme détermine bien tous les facteurs cyclotomiques généralisés de  $F(\underline{x})$ . Soit  $m \in \mathbb{N}$  et  $\beta \in \mathbb{Z}^n$  tels que

$$J\Phi_m(\underline{x}^\beta) | F(\underline{x}).$$

D'après les lemmes 2.1 et 2.2, le vecteur  $\beta$  fera bien partie de l'ensemble  $V$  déterminé à l'étape 2 de l'algorithme. Supposons maintenant que ce vecteur est choisi à l'étape 3. Comme  $J\Phi_m(\underline{x}^\beta) | F(\underline{x})$ , le lemme 2.3 garantit que  $m$  appartient bien à chacun des ensembles  $S_f$  déterminés à l'étape 4 et donc à l'ensemble  $S_\beta$  de l'étape 5. Finalement, le couple  $(\beta, m)$  appartient bien à l'ensemble  $S_F$  renvoyé par l'algorithme.

Réciproquement, le lemme 2.3 assure que tous les couples retournés par l'algorithme correspondent bien à des facteurs cyclotomiques généralisés du polynôme  $F$ .

### 2.4 Complexité

Estimons maintenant le nombre d'opérations binaires nécessaires à l'exécution de chaque étape.

1. Cette première étape ne nécessite pas de calcul.
2. On a déjà vu au lemme 2.2 que cette étape nécessite

$$O_N(n \log d (\log \log d)^2 \log \log \log d)$$

opérations binaires. On note également que l'on obtient au plus  $N-1 = O_N(1)$  vecteurs.

3. Pour former le système du lemme de Kaltofen et Koiran, on doit substituer à  $x_n$  un monôme en  $x_1, \dots, x_{n-1}$  puis regrouper les monômes de même degré. Tout ceci se fait en au plus

$$O_N(n \log d \log \log d \log \log \log d + h)$$

opérations binaires. On peut noter aussi que les polynômes qui composent ce système sont de degré au plus  $d^2$ , de hauteur au plus  $h$  et possèdent au plus  $N$  termes non nuls.

4. Le système obtenu se compose de  $O_N(1)$  polynômes. On fixe  $f$  un de ces polynômes et on applique l'algorithme 2.4 pour déterminer  $S_f$ . Ceci se fait en

$$O_N(\log d (\log \log d)^2 \log \log \log d + h)$$

opérations binaires. Réaliser cette étape pour les  $O_N(1)$  polynômes nécessite donc  $O_N(\log d (\log \log d)^2 \log \log \log d + h)$  opérations binaires.

5. Dans [FGS08], les auteurs montrent que  $\#S_f = O_N(1)$  donc pour déterminer  $S_{\beta}$  on effectue des calculs de pgcd et ppcm sur  $O_N(1)$  entiers majorés par  $d^2$ . Tout ceci se fait donc en  $O_N(\log d (\log \log d)^2 \log \log \log d)$  opérations binaires.

On obtient finalement une complexité de

$$O_N(n \log d (\log \log d)^2 \log \log \log d + h)$$

pour l'ensemble de l'algorithme, ce qui était annoncé dans le théorème 1.2.  $\square$

### 3 Points de torsion

On va, dans cette partie, démontrer le théorème 1.1. Comme les multiplications d'entiers et les calculs de pgcd n'affectent pas la complexité de l'algorithme de ce théorème, on utilisera, pour alléger les calculs, la borne  $O(s^2)$  pour le nombre d'opérations binaires nécessaires au calcul du produit

de deux entiers de taille  $s$  et pour le nombre d'opérations binaires nécessaires au calcul du pgcd de deux entiers de taille  $s$ .

Donnons nous un polynôme  $F(x, y) \in \mathbb{Z}[x, y]$  de degré  $d$ , de hauteur  $h$  et possédant au plus  $N$  termes non nuls. On cherche alors à déterminer s'il existe un point de torsion en lequel s'annule le polynôme  $F$ . Rappelons que l'on va distinguer deux cas, selon que le polynôme possède un facteur cyclotomique généralisé ou non. Si c'est le cas, on peut déterminer un tel facteur grâce à l'algorithme du théorème 1.2 puis déterminer un point de torsion en lequel s'annule ce facteur. En effet, si

$$J\Phi_m(x^{\gamma_1}y^{\gamma_2})|F(x, y),$$

où  $\gamma_1, \gamma_2$  sont deux entiers premiers entre eux, le polynôme  $F$  s'annule alors en  $(\zeta_m^u, \zeta_m^v)$  où  $\zeta_m$  est une racine de l'unité d'ordre  $m$  et  $u, v$  sont des entiers tels que

$$u\gamma_1 + v\gamma_2 = 1.$$

De plus,  $u$  et  $v$  peuvent être déterminés grâce à l'algorithme d'Euclide. Pour effectuer ces opérations, on doit appliquer l'algorithme du théorème 1.2 ce qui nécessite  $O_N(\log d(\log \log d)^2 \log \log \log d + h(F))$  opérations binaires, puis appliquer l'algorithme d'Euclide aux entiers  $\gamma_1$  et  $\gamma_2$ . Cette dernière opération nécessite  $O(\log^2 d)$  opérations binaires car

$$\log(\gamma_i) = O(\log d), \quad \text{pour } i = 1, 2$$

puisque  $J\Phi_m(x^{\gamma_1}y^{\gamma_2})|F(x, y)$ .

On peut donc maintenant supposer que  $F$  ne possède pas de facteur cyclotomique. Cette hypothèse nous permet de majorer le degré des points de torsion en lesquels s'annule le polynôme  $F$  comme nous l'avons vu dans l'introduction. Pour limiter le nombre de tests à effectuer, on va montrer que si le polynôme  $F$  s'annule en un point de  $\mu_\infty^2$ , il en existe un dont on peut majorer les facteurs premiers de l'ordre dans le groupe  $\mu_\infty^2$ .

### 3.1 Majoration des facteurs premiers

Pour  $e \in \mathbb{N}$ , on note  $P_e$  l'ensemble des premiers divisant  $e$  et pour  $p$  premier et  $m \in \mathbb{N}$ , on note  $\nu_p(m)$  l'exacte puissance de  $p$  qui divise  $m$ .

**Lemme 3.1** *Soient  $e, m \geq 1$  et  $\omega \in \mu_e$ , alors il existe  $f \geq 1$  et  $\zeta \in \mu_f$  tels que  $P_f = P_e$  et  $\zeta^m = \omega$ .*

**Démonstration :** On pose

$$f = \prod_{p|e} p^{\nu_p(e)+\nu_p(m)}.$$

On a  $P_f = P_e$  et on vérifie facilement que l'application  $\mu_f \rightarrow \mu_e, \zeta \mapsto \zeta^m$  est surjective.  $\square$

Etendons maintenant ce résultat en dimension 2. Soit  $G$  un groupe et  $g \in G$ , on note  $\text{ord}(g)$  l'ordre de  $g$  dans  $G$ .

**Lemme 3.2** Soient  $\omega \in \mu_\infty^2$  et  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2(\mathbb{Z})$  tel que  $\det(A) \neq 0$ . Alors il existe  $\zeta \in \mu_\infty^2$  tel que  $P_{\text{ord}(\zeta)} \subseteq P_{\text{ord}(\omega)}$  et

$$\zeta_1^a \zeta_2^b = \omega_1 \quad , \quad \zeta_1^c \zeta_2^d = \omega_2. \quad (1)$$

**Démonstration :** En faisant des opérations inversibles, on vérifie que le système (1) est équivalent à un système de la forme

$$\zeta_1^{a'} \zeta_2^{b'} = \omega'_1 \quad , \quad \zeta_2^{d'} = \omega'_2 \quad (2)$$

pour certains  $\omega'_i \in \mu_\infty$  et  $a', b', d' \in \mathbb{Z}$  avec  $a'd' \neq 0$ . On a  $\text{ord}(\omega'_2) | \text{ord}(\omega)$  et grâce au lemme 3.1 on peut trouver  $\zeta_2$  tel que  $P_{\text{ord}(\zeta_2)} \subseteq P_{\text{ord}(\omega)}$  et  $\zeta_2^{d'} = \omega'_2$ . On peut donc récrire la première équation du système comme

$$\zeta_1^{a'} = \omega'_1 \zeta_2^{-b'}.$$

On a  $\text{ord}(\omega'_1 \zeta_2^{-b'}) = \text{ppcm}(\text{ord}(\omega'_1), \text{ord}(\zeta_2^{-b'}))$  et donc  $\text{ord}(\omega'_1 \zeta_2^{-b'}) | \text{ord}(\omega)$ . On peut alors appliquer à nouveau le lemme 3.1 pour trouver une solution  $\zeta_1$  de cette équation telle que  $P_{\text{ord}(\zeta_1)} \subseteq P_{\text{ord}(\omega)}$  et le résultat est démontré.  $\square$

On note pour  $m \in \mathbb{N}$

$$\Psi(m) = 2 + \sum_{p|m} (p - 2).$$

En utilisant l'idée du lemme 3.1 et du théorème 5 de [CJ76], Filaseta et Schinzel montrent le résultat suivant (voir [FS04], corollaire 1).

**Proposition 3.3** Soit  $f \in \mathbb{Z}[x]$  ayant au plus  $N$  termes non nuls. Si  $f$  possède un facteur cyclotomique, alors il existe un entier positif  $m$  tel que

$$\Psi(m) \leq N \quad \text{et} \quad \Phi_m(x) | f(x).$$

Pour  $F(x, y) = \sum_{i=1}^N a_i x^{\alpha_i} y^{\beta_i}$ , on note  $L_F$  le  $\mathbb{Z}$ -module :

$$L_F = \sum_{1 \leq j, k \leq N} \mathbb{Z}((\alpha_j, \beta_j) - (\alpha_k, \beta_k)).$$

On peut alors généraliser cette proposition à l'aide du lemme 3.2.

**Proposition 3.4** Soit  $F(x, y) \in \mathbb{Z}[x, y]$  un polynôme possédant au plus  $N$  termes non nuls. Si  $F(x, y) = 0$  admet une solution dans  $\mu_\infty^2$ , il existe  $\zeta \in \mu_\infty^2$  tel que :

$$F(\zeta) = 0 \quad \text{et} \quad \Psi(\text{ord}(\zeta)) \leq N.$$

**Démonstration :** Si  $N(F)$  est de dimension 1, le polynôme  $F(x, y)$  se ramène à un polynôme en une variable via un changement monomial et ce cas est alors couvert par le corollaire 1 de [FS04]. Supposons donc  $N(F)$  de dimension 2 et soit  $\eta \in \mu_\infty^2$  telle que  $F(\eta) = 0$ . On écrit  $F = \sum_k F_k$  tel que  $F_k(\eta) = 0$  soit une somme minimale pour chaque  $k$ , c'est à dire qu'aucune somme extraite de  $F_k(\eta)$  ne s'annule. Posons alors

$$L = \sum_k L_{F_k}$$

qui est bien un sous-module libre de  $\mathbb{Z}^2$  de rang 2 (car  $N(F)$  est de dimension 2) et on a  $L \supset L_F$ . Soient  $(a, b), (c, d) \in \mathbb{Z}^2$  une base de  $L$  de sorte que  $F_k(x, y) = H_k(x^a y^b, x^c y^d)$  pour certains  $H_k \in \mathbb{Z}[x, y]$ . Si on pose  $\omega = (\eta_1^a \eta_2^b, \eta_1^c \eta_2^d) \in \mu_\infty^2$ , le théorème 5 de [CJ76] entraîne alors que

$$\Psi(\text{ord}(\omega)) \leq \max_k (\text{nombre de termes de } H_k) \leq N$$

et on conclut par application du lemme 3.2. □

## 3.2 Algorithmes auxiliaires

La procédure principale du théorème 1.1 fera appel aux algorithmes décrits dans cette partie. Le premier d'entre eux permet de déterminer les entiers

satisfaisant la condition de la proposition 3.4 et la condition donnée dans l'introduction portant sur le degré. Il est décrit dans le lemme suivant et est largement inspiré de l'algorithme B de [FS04]. On note dans la suite  $\varphi$  la fonction indicatrice d'Euler.

**Lemme 3.5** *Il existe un algorithme qui a la propriété suivante : étant donnés  $N, D \in \mathbb{N}$ , l'algorithme détermine*

$$W = \{m \in \mathbb{N} : \Psi(m) \leq N \text{ et } \varphi(m) \leq D\}$$

où les entiers qui composent cet ensemble sont donnés par leur factorisation en produit de facteurs premiers en

$$O_N \left( (\log D)^{\kappa \sqrt{\frac{N}{\log N}}} \right)$$

opérations binaires, où  $\kappa$  est un réel strictement positif effectivement calculable.

**Démonstration :** L'algorithme est le suivant :

**Entrée :** Deux entiers  $N$  et  $D$ .

**Sortie :** L'ensemble  $\{m \in \mathbb{N} : \Psi(m) \leq N \text{ et } \varphi(m) \leq D\}$ .

1. Faire  $W \leftarrow \emptyset$  et déterminer l'ensemble  $P_N = \{p_1, \dots, p_r\}$  des nombres premiers inférieurs à  $N$ .
2. Déterminer l'ensemble  $Q$  des sous-ensembles  $\{q_1, \dots, q_s\}$  de  $P_N$  pour lesquels  $2 + \sum_{j=1}^s (q_j - 2) \leq N$ .
3. Pour chaque  $\{q_1, \dots, q_s\} \in Q$ , former l'ensemble des listes  $((q_1, e_1), \dots, (q_s, e_s))$  telles que pour  $1 \leq j \leq s$ , on a  $1 \leq e_j \leq \left\lceil \frac{\log D}{\log q_j} \right\rceil + 1$  et ajouter à  $W$  les listes qui vérifient la condition supplémentaire

$$\prod_{j=1}^s q_j^{e_j - 1} (q_j - 1) \leq D.$$

4. Renvoyer  $W$  et terminer.

Démontrons que l'algorithme détermine bien l'ensemble voulu.  
 Seule l'étape 3 mérite des explications : soit  $m \in W$ ,  $p$  un nombre premier divisant  $m$  et  $e = \nu_p(m)$ . Alors

$$D \geq \varphi(m) \geq \varphi(p^e) = p^{e-1}(p-1) \geq p^{e-1}$$

et on a bien

$$e \leq \left\lceil \frac{\log D}{\log p} \right\rceil + 1.$$

Il est alors clair que l'ensemble renvoyé par l'algorithme est bien celui recherché. Il ne nous reste qu'à déterminer la complexité de cet algorithme.

1. Cette étape peut être réalisée à l'aide du crible d'Ératosthène en  $O_N(1)$  opérations binaires.
2. Déterminons tout d'abord une majoration du cardinal de  $Q$ . Soit  $(q_1, \dots, q_s) \in Q$ . On a alors

$$N \geq 2 + \sum_{j=1}^s (q_j - 2) \geq \sum_{j=1}^s (p_j - 2).$$

Or, d'après le théorème des nombres premiers, on a

$$\sum_{p \leq x} p \sim \frac{x^2}{2 \log x}$$

donc  $\sum_{j=1}^s (p_j - 2) \geq cs^2 \log s$ , où  $c$  est un réel strictement positif. On obtient finalement

$$s \leq C \sqrt{\frac{N}{\log N}}, \quad \text{où } C > 0.$$

On appelle dans la suite  $K(N)$  cette borne pour  $s$ . Comme il y a  $r$  choix possibles pour chaque premier  $q_j$  dans un ensemble de  $Q$ , on en déduit la majoration suivante du cardinal de  $Q$  :

$$\#Q \leq r^{K(N)} \leq N^{K(N)} \leq e^{C\sqrt{N \log N}}.$$

On peut ensuite déterminer les éléments de  $Q$  en formant, pour  $s$  fixé, les sous ensembles de  $P_N$  possédant  $s$  éléments. Il y en a au plus  $r^s$  et vérifier, pour chacun d'eux, la condition

$$2 + \sum_{j=1}^s (q_j - 2) \leq N$$

peut se faire en  $O_N(1)$  opérations binaires. Comme  $1 \leq s \leq K(N)$ , cette étape nécessite  $O_N(1)$  opérations binaires.

3. Déterminons maintenant une majoration du nombre de listes formées à cette étape. On observe tout d'abord que  $\left\lceil \frac{\log D}{\log 2} \right\rceil + 1 \leq 1 + 2 \log D$  et que  $\left\lceil \frac{\log D}{\log p_j} \right\rceil + 1 \leq 1 + \log D$  pour  $j > 1$ . Ainsi, on forme à partir de chaque élément  $\{q_1, \dots, q_s\}$  de  $Q$ , au plus  $2(1 + \log D)^{K(N)}$  listes. Comme il y a au plus  $r^{K(N)}$  éléments dans  $Q$ , on obtient au plus

$$O_N \left( 2r^{K(N)} (1 + \log D)^{K(N)} \right) = O_N \left( (\log D)^{C \sqrt{\frac{N}{\log N}}} \right)$$

listes.

Evaluons maintenant le temps nécessaire à la vérification de la condition

$$\prod_{j=1}^s q_j^{e_j - 1} (q_j - 1) \leq D.$$

Pour calculer  $q_j^{e_j - 1} (q_j - 1)$ , on effectue récursivement  $e_j = O(\log D)$  multiplications d'entiers majorés par  $N$  et comme le résultat n'excède jamais  $D$  par définition de  $e_j$ , ceci nécessite  $O_N(\log^3 D)$  opérations binaires. On doit ensuite effectuer  $s$  multiplications pour obtenir le nombre  $\prod_{j=1}^s q_j^{e_j - 1} (q_j - 1)$ . De même, comme à chaque étape le résultat ne doit pas excéder  $D$ , ceci nécessite  $O_N(s \log^2 D) = O_N(\log^2 D)$  opérations binaires et la vérification de l'inégalité nécessite  $O(\log D)$  opérations binaires. Au total, la complexité de cette étape est

$$O_N \left( (\log D)^3 (\log D)^{C \sqrt{\frac{N}{\log N}}} \right) = O_N \left( (\log D)^{C' \sqrt{\frac{N}{\log N}}} \right),$$

où  $C' > 0$ .

4. On doit, à cette étape, écrire tous les éléments de  $W$ . On peut réutiliser la borne  $O_N \left( (\log D)^{C \sqrt{\frac{N}{\log N}}} \right)$  pour le cardinal de cet ensemble et comme écrire une de ces listes nécessite  $O_N(\log \log D)$  opérations binaires, cette étape a pour complexité :

$$O_N \left( (\log D)^{C'' \sqrt{\frac{N}{\log N}}} \right), \quad \text{où } C'' > 0.$$

Finalement, l'exécution de cet algorithme nécessite  $O_N \left( (\log D)^{\kappa \sqrt{\frac{N}{\log N}}} \right)$  opérations binaires, où  $\kappa$  est un réel strictement positif que l'on peut calculer explicitement. On ne dépasse donc pas la complexité annoncée.  $\square$

On aura également besoin de résoudre des systèmes linéaires et nous utiliserons pour cela les deux lemmes suivants :

**Lemme 3.6** *Il existe un algorithme qui a la propriété suivante : étant donnés trois entiers  $a, b$  et  $m$  tels que  $0 \leq a < m$  et  $0 \leq b < m$ , l'algorithme détermine si l'ensemble  $S$  des solutions de*

$$ax = b \pmod{m}$$

*est vide en au plus  $O(\log^2 m)$  opérations binaires. De plus, dans le cas où  $S$  est non vide, l'algorithme renvoie le couple  $(e, d) \in \mathbb{N}^*$  tel que  $d|m$  et*

$$S = \left\{ e + f \frac{m}{d} \pmod{m} : 0 \leq f < d \right\}.$$

**Démonstration :** L'algorithme est le suivant :

**Entrée :** Trois entiers  $a, b, m$ .

**Sortie :** Deux entiers  $e$  et  $d$  tels que les solutions de l'équation  $ax = b \pmod{m}$  sont données par  $\{e + f \frac{m}{d} \pmod{m} : 0 \leq f < d\}$ .

1.  $d \leftarrow \text{pgcd}(a, m)$ .
2. Si  $d \nmid b$  renvoyer « l'équation n'a pas de solution » et terminer, sinon  $A \leftarrow \frac{a}{d}, B \leftarrow \frac{b}{d}$  et  $M \leftarrow \frac{m}{d}$ .
3.  $e \leftarrow A^{-1}B \pmod{M}$ , renvoyer  $(e, d)$  et terminer.

Il est clair que cet algorithme détermine bien les solutions recherchées. Déterminons le nombre d'opérations binaires nécessaires à son exécution.

L'étape 1 est un calcul de pgcd sur deux entiers majorés par  $m$  et ceci nécessite  $O(\log^2 m)$  opérations binaires.

L'étape 2 nécessite  $O(\log^2 m)$  opérations binaires puisqu'on effectue quatre divisions sur des entiers majorés par  $m$ .

L'étape 3 nécessite également  $O(\log^2 m)$  opérations binaires puisqu'on effectue une inversion modulo un entier majoré par  $m$  et une multiplication modulo le même entier.

Au total, on effectue donc bien  $O(\log^2 m)$  opérations binaires.  $\square$

**Lemme 3.7** *Il existe un algorithme qui a la propriété suivante : étant donnés deux entiers  $m$ ,  $K$ , et des triplets  $(a_i, b_i, c_i)_{1 \leq i \leq K}$  l'algorithme détermine si le système*

$$a_i x + b_i y = c_i \pmod{m} \quad 1 \leq i \leq K$$

*possède une solution en au plus  $O(K \log^2 m)$  opérations binaires. De plus, l'algorithme retourne une solution  $(x, y)$  quand ce système en possède une.*

**Démonstration :** L'algorithme effectue les étapes suivantes :

1. (a)  $a \leftarrow \text{pgcd}(m, \{a_i\}_{1 \leq i \leq K})$  et déterminer  $\{u_i\}_{1 \leq i \leq K}$  à l'aide de l'algorithme d'Euclide étendu de sorte que  $a = \sum_{i=1}^K a_i u_i$ .
- (b)  $(a, b, c) \leftarrow (a, \sum_{i=1}^K b_i u_i \pmod{m}, \sum_{i=1}^K c_i u_i \pmod{m})$ .
- (c) Pour  $1 \leq i \leq K$ ,

$$(0, B_i, C_i) \leftarrow (0, b_i - b \frac{a_i}{a} \pmod{m}, c_i - c \frac{a_i}{a} \pmod{m})$$

2. (a)  $B \leftarrow \text{pgcd}(m, \{B_i\}_{1 \leq i \leq K})$  et déterminer  $\{U_i\}_{1 \leq i \leq K}$  à l'aide de l'algorithme d'Euclide étendu de sorte que

$$B = \sum_{i=1}^K B_i U_i \pmod{m}.$$

- (b)  $C \leftarrow \sum_{i=1}^K U_i C_i \pmod{m}$ .
- (c) Si  $\forall 1 \leq i \leq K, C \frac{B_i}{B} = C_i$ , aller à l'étape suivante sinon renvoyer « le système n'a pas de solution » et terminer.

3. (a) Si l'équation  $By = C \pmod m$  n'a pas de solution, renvoyer « le système n'a pas de solution » et terminer, sinon déterminer deux entiers  $E$  et  $D$  tels que les solutions de cette équation soit données par  $S = \{E + z\frac{m}{D} \pmod m \mid 0 \leq z < D\}$ .
- (b) Résoudre l'équation  $b\frac{m}{D}z = c - bE \pmod a$ . Si cette équation n'a pas de solution telle que  $0 \leq z < D$ , renvoyer « le système n'a pas de solution » et terminer, sinon choisir une solution  $z_0$  de cette dernière équation vérifiant  $0 \leq z_0 < D$ .
- (c) Renvoyer le couple  $(\frac{D(c-bE)-bmz_0}{Da}, E + z_0\frac{m}{D})$  et terminer.

Il est clair que l'algorithme résout bien le système d'équation. Evaluons donc le coût de l'exécution de chaque étape. Les étapes 1 et 2 nécessitent  $O(K \log^2 m)$  opérations binaires puisqu'on y effectue un nombre linéaire en  $K$  d'opérations de base  $(+, -, /, \times)$  et des calculs de pgcd sur des entiers majorés par  $m$ .

A l'étape 3, on fait deux fois appel à l'algorithme du lemme précédent, ce qui nécessite  $O(\log^2 m)$  opérations binaires puis on effectue encore des opérations de bases sur des entiers majorés par  $m$ . Cette étape nécessite donc  $O(\log^2 m)$  opérations binaires.

Au total, on ne dépasse donc pas la complexité annoncée.  $\square$

### 3.3 Démonstration du théorème 1.1

On va tout d'abord décrire l'algorithme, puis prouver sa correction et enfin évaluer sa complexité.

#### 3.3.1 Algorithme

On se donne ici un polynôme  $F(x, y) = \sum_{i=1}^N a_i x^{\alpha_i} y^{\beta_i} \in \mathbb{Z}[x, y]$  de degré  $d$ , de hauteur  $h$  et ne possédant pas de facteur cyclotomique généralisé. On cherche à déterminer un point de torsion en lequel s'annule ce polynôme et on effectue pour cela les opérations suivantes :

**Entrée :** Un polynôme  $F(x, y) = \sum_{i=1}^N a_i x^{\alpha_i} y^{\beta_i} \in \mathbb{Z}[x, y]$ .

**Sortie :** L'ensemble vide si  $F$  ne s'annule en aucun point de torsion et un triplet  $(m, a, b)$  d'entiers tel que  $F$  s'annule en  $(\zeta_m^a, \zeta_m^b)$ , où  $\zeta_m$  est une racine

primitive  $m$ -ième de l'unité, si  $F$  s'annule en un point de torsion.

1. Faire  $W \leftarrow \{m \in \mathbb{N}, \Psi(m) \leq N \text{ et } \varphi(m) \leq 11d^2\}$ .

2. **Si**  $W = \emptyset$  **aller** à l'étape 3, **sinon** choisir  $m \in W$  tel que  $m = \prod_{j=1}^s p_j^{e_j}$ ,  
faire  $W \leftarrow W \setminus \{m\}$  **puis** faire :

(a) Calculer  $F(t^a, t^b) \times \prod_{p|m} \left(t^{\frac{m}{p}} - 1\right) = \sum_{l=1}^M b_l t^{\lambda_l(a,b)}$ .

(b) Faire  $S \leftarrow \{\{S_1, \dots, S_r\} : \cup_{k=1}^r S_k = \{1, \dots, M\}, \forall k \neq k' S_k \cap S_{k'} = \emptyset, \text{ et } \forall 1 \leq k \leq r, \sum_{l \in S_k} b_l = 0\}$ .

i. **Si**  $S = \emptyset$  **aller** à l'étape 2, **sinon** choisir  $\{S_1, \dots, S_r\} \in S$  et faire  $S \leftarrow S \setminus \{\{S_1, \dots, S_r\}\}$ .

ii. Résoudre le système  $\lambda_l(a, b) = \lambda_{l'}(a, b) \pmod m \forall l, l' \in S_k \forall k \in \{1, \dots, r\}$ .

iii. **Si** le précédent système possède une solution  $(a, b)$ , **renvoyer**  $(m, a, b)$  et terminer **sinon retourner** à l'étape 2(b)i.

3. **Renvoyer** « le polynôme ne s'annule pas en un point de torsion » et terminer.

### 3.3.2 Correction

Supposons dans un premier temps que  $F$  s'annule en un point de torsion et montrons que l'algorithme en détermine bien un. D'après la proposition 3.4, le polynôme  $F$  s'annule en un point de torsion dont l'ordre  $m$  vérifie  $\Psi(m) \leq N$ . On note dans la suite  $(\zeta_m^a, \zeta_m^b)$  un tel zéro. De plus, on a vu que

$$\varphi(m) = [\mathbb{Q}(\zeta_m) : \mathbb{Q}] \leq 11d^2,$$

donc  $m$  appartient bien à l'ensemble  $W$  déterminé à l'étape 1. Supposons maintenant que  $m$  est choisi à l'étape 2. Comme  $F(\zeta_m^a, \zeta_m^b) = 0$ , on a

$$\Phi_m(t) | F(t^a, t^b)$$

et donc  $t^m - 1$  divise

$$F(t^a, t^b) \times \prod_{p|m} \left(t^{\frac{m}{p}} - 1\right) = \sum_{l=1}^M b_l t^{\lambda_l(a,b)}.$$

Posons maintenant

$$S_k = \{1 \leq l \leq M \mid \lambda_l(a, b) = k \pmod{m}\} \text{ pour } 0 \leq k \leq m-1.$$

Alors les  $S_k$  forment une partition de  $\{1, \dots, M\}$  et comme  $\sum_{l=1}^M b_l t^{\lambda_l(a,b)}$  est nul modulo  $t^m - 1$ , on a bien

$$\sum_{l \in S_k} b_l = 0, \quad \forall 0 \leq k \leq m-1.$$

Ainsi,  $\{S_0, \dots, S_{m-1}\}$  est un élément de l'ensemble  $S$  déterminé à l'étape 2b et si celui-ci est choisi à l'étape 2(b)i, le couple  $(a, b)$  sera une solution du système formé à l'étape 2(b)ii et l'algorithme renverra donc bien une solution. Supposons maintenant que l'algorithme renvoie  $(m, a, b)$  et montrons que  $F(\zeta_m^a, \zeta_m^b) = 0$ . Dans ce cas on a

$$F(t^a, t^b) \times \prod_{p \mid m} \left( t^{\frac{m}{p}} - 1 \right) \equiv 0 \pmod{t^m - 1}$$

et on a alors

$$\Phi_m(t) \mid F(t^a, t^b).$$

Mais ceci signifie exactement que  $F(\zeta_m^a, \zeta_m^b) = 0$  et l'algorithme décrit en 3.3.1 est donc correct.

### 3.3.3 Complexité

1. D'après le lemme 3.5, cette étape nécessite

$$O_N \left( (\log d)^{\kappa \sqrt{\frac{N}{\log N}}} \right)$$

opérations binaires.

2. (a) Soit  $m = \prod_{j=1}^s q_j^{e_j}$  un élément de l'ensemble  $W$  déterminé à l'étape précédente. Comme on a d'une part  $\varphi(m) \leq 11d^2$  et d'autre part l'inégalité classique  $\sqrt{\frac{m}{2}} \leq \varphi(m)$ , on a finalement  $m \leq 242d^4$  et on a ainsi

$$\log m = O(\log d).$$

Une fois fixé l'entier  $m$ , on calcule récursivement le produit

$$F(t^a, t^b) \times \prod_{j=1}^s \left( t^{\frac{m}{q_j}} - 1 \right)$$

où  $a$  et  $b$  sont des inconnues à déterminer. A chacune des  $s$  multiplications par un facteur  $t^{\frac{m}{q_j}} - 1$ , chaque monôme du polynôme en produit deux de la façon suivante :

$$at^\alpha \rightarrow at^{\alpha + \frac{m}{q_j}} - at^\alpha.$$

Comme à chaque étape le polynôme possède  $O_N(1)$  termes, ce calcul nécessite  $O_N(\log m + h)$  opérations binaires. Notons que le polynôme obtenu possède  $M = O_N(1)$  termes et chacun de ses coefficients est majoré en valeur absolue par  $O_N(h)$ .

- (b) Comme  $M = O_N(1)$ , déterminer toutes les partitions de  $\{1, \dots, M\}$  nécessite  $O_N(1)$  opérations binaires. De plus, pour chaque partition  $S_1, \dots, S_k$  de  $\{1, \dots, M\}$ , vérifier la condition  $\forall 1 \leq k \leq r, \sum_{l \in S_k} b_l = 0$  nécessite  $O_N(\log h)$  opérations binaires.
  - i. Cette étape ne nécessite que  $O(1)$  opérations binaires.
  - ii. On résout le système grâce à l'algorithme du lemme 3.7 et ceci nécessite  $O_N(\log^2 m)$  opérations binaires car ce système est constitué de  $O_N(1)$  équations.
  - iii. Cette opération ne nécessite que  $O_N(\log m)$  opérations binaires pour écrire le résultat.

- 3. Cette dernière étape ne nécessite que  $O(1)$  opérations binaires.

Au total, on effectue donc bien  $O_N \left( h(\log d)^\delta \sqrt{\frac{N}{\log N}} \right)$  opérations binaires, où  $\delta$  est un réel strictement positif effectivement calculable.  $\square$

## References

- [BS02] F. Beukers, C. Smyth, *Cyclotomic points on curves*, Number theory for the millenium, **1** (Urbana, Il., 2000), 67 – 85, 2002.
- [CJ76] J. H. Conway et A. J. Jones, *Trigonometric Diophantine equations (On vanishing sums of roots of unity)*, Acta Arith. 30 (1976), 229 – 240.
- [FGS08] M. Filaseta, A. Granville et A. Schinzel, *Irreducibility and Greatest Common Divisor Algorithms for Sparse Polynomials*, Number Theory and Polynomials, London Math. Soc. lecture notes,
- [FS04] M. Filaseta et A. Schinzel, *On testing the divisibility of lacunary polynomials by cyclotomic polynomials*, Math. Comp. 73 2004, 957 – 965.
- [K70] D. E. Knuth, *The analysis of algorithms*, Actes du Congrès International des Mathématiciens, 1970, 269 – 274.
- [KK06] E. Kaltofen et P. Koiran, *Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields*, ISAAC 2006, Proc. 2006 Internat. Symp. Symbolic Algebraic comput. 2006, 162 – 168.
- [La83] S. Lang, *Fundamentals of diophantine geometry*, Springer-Verlag, 1983.
- [SS71] A. Schönhage et V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing vol. 7, 1971, 281 – 292.