Expose de maîtrise
Factorisation dans $\mathbb{Z}[X]$

Abuaf Roland
Boyer Ivan

Sujet proposé par François Loeser

20 juin 2007

Table des matières

1 Précisions sur la complexité

Définition 1.1 *Un algorithme sur les entiers sera dit* polynomial en la taille des données *si le nombre d'opérations arithmétiques élémentaires est dominé par un polynôme en le logarithme du plus grand entier des données.*

Cette définition est raisonnable car un entier n est codé en base 2 et occupe donc un espace de $\log_2(n)$ en mémoire.

La majeure partie de cet exposé s'attache à montrer que le problème de la factorisation des polynômes à coefficients entiers est dans la classe **P**, c'est-à-dire qu'il existe un algorithme qui résout ce problème en temps polynomial (par rapport aux entiers des coefficients et au degré du polynôme)

Il ne s'agit pas de l'algorithme le plus efficace dans les cas courants et c'est pourquoi nous nous attacherons principalement au caractère polynomial de la complexité et non pas à l'optimalité.

2 Algorithme de Berlekamp

2.1 Lemme de Berlekamp

Dans cette section, nous allons considérer un algorithme de factorisation des polynômes à coefficients entiers dans les corps finis, celui de Berlekamp. C'est certainement la méthode la plus connue pour les corps finis car son implémentation est relativement aisée et son coût est polynomial en le degré du polynôme.

Plus précisément, il s'agit d'un algorithme de factorisation de polynômes sans facteurs carrés, dans $\mathbb{F}_p[X]$, pour p premier. En fait, l'idée générale marche pour tous les corps finis, mais les calculs de base (addition, multiplication, inverse) sont plus faciles dans $\mathbb{F}_p$.

Nous allons rappeler deux lemmes d'algèbre élémentaire, qui nous permettront d'établir le principe fondamental de l'algorithme. Dans la suite, on notera p un nombre premier.

Proposition 2.1.1 (Lemme Chinois) *Soit A un anneau principal et $x \in A$ avec $x = \prod_{i=1}^{k} p_i^{\alpha_i}$ sa décomposition en produit d'irréductibles, alors on a l'isomorphisme suivant :*

$$\begin{array}{ccc} A/(x) & \xrightarrow{\sim} & \prod_{i=1}^{k} A/(p_i^{\alpha_i}) \\ \overline{a}^x & \longmapsto & \left(\overline{a}^{p_1^{\alpha_1}}, \ldots, \overline{a}^{p_k^{\alpha_k}}\right). \end{array}$$

Proposition 2.1.2 (Lemme de Rupture) *Si f est un polynôme irréductible sur $\mathbb{F}_p$ alors le quotient $\mathbb{F}_p[X]/(f)$ est une extension finie de corps de $\mathbb{F}_p$ dans laquelle f admet une racine. On l'appelle corps de rupture de f sur $\mathbb{F}_p$.*

On en déduit le résultat suivant :

Soit f un polynôme à coefficients entiers sans facteurs carrés, alors $\mathbb{F}_p[X]/(f)$ est isomorphe à un produit direct de corps finis.

C'est une conséquence évidente des lemmes qui précèdent. Notons que l'isomorphisme chinois est un isomorphisme d'algèbre. Nous allons maintenant énoncer et démontrer le lemme de Berlekamp qui donne une expression intéressante du polynôme à factoriser.

Proposition 2.1.3 (Lemme de Berlekamp) *Soit f un polynôme sur $\mathbb{F}_p$ sans facteurs carrés, on note A l'algèbre $\mathbb{F}_p[X]/(f)$ et S l'endomorphisme de A comme espace vectoriel sur $\mathbb{F}_p$:*

$$\begin{aligned} S : A &\to A \\ a &\mapsto a^p - a. \end{aligned}$$

Si $t \in \ker S$ alors on a l'égalité suivante :

$$f = \prod_{\alpha \in \mathbb{F}_p} \operatorname{pgcd}(f, t - \alpha).$$

Preuve :
► On peut supposer $f \neq 0$. Grâce à l'isomorphisme chinois, on va pouvoir travailler sur le produit de corps pour déterminer le noyau de S. En effet si $t \in \ker S$ alors chaque composante t_i de t (vu comme vecteur du produit de corps) vérifie l'équation $t_i^p - t_i = 0$ dans une extension de $\mathbb{F}_p$. Or par définition même de $\mathbb{F}_p$, cela signifie que $t_i \in \mathbb{F}_p$. Dès lors, dans l'algèbre produit, $\ker S$ est le produit des $\mathbb{F}_p$.

Ainsi si on pose $f = \prod f_i$ où les f_i sont irréductibles sur $\mathbb{F}_p$ et tous distincts (décomposition sans facteurs carrés), $t - \alpha$ a pour $i^{\text{ème}}$ composante $t_i - \alpha$ dans l'algèbre produit. Donc si $\alpha = t_i$ alors $t - \alpha$ est divisible par f_i. Ceci prouve que f divise $\prod_{\alpha \in \mathbb{F}_p} \operatorname{pgcd}(f, t - \alpha)$.

D'un autre côté une écriture de Bézout montre immédiatement que les $t - \alpha,\ \alpha \in \mathbb{F}_p$ sont deux à deux premiers entre eux. Ainsi les $\operatorname{pgcd}(f, t - \alpha),\ \alpha \in \mathbb{F}_p$ sont deux à deux premiers entre eux et divisent f et donc $\prod_{\alpha \in \mathbb{F}_p} \operatorname{pgcd}(f, t - \alpha)$ divise f. ◄

Notons que nous avons confondu t et un de ses représentants dans A (disons qu'à chaque fois que l'on écrivait $\operatorname{pgcd}(f, t - \alpha)$, on évoquait un représentant de t). Le fait que t soit de degré strictement inférieur à f montre que, si on peut trouver t de degré supérieur à un, alors l'un des pgcd au moins sera un diviseur de f distinct de f et de degré supérieur à un.

2.2 Noyau de Berlekamp

La question se pose maintenant de savoir si l'on peut trouver un élément du noyau de S qui ne soit pas une constante. La démonstration du lemme de Berlekamp fait apparaître un résultat dans cette direction :

Porisme 2.2.1 *Si $f = f_1 \dots f_r$ est la décomposition sans facteurs carrés de f, alors la dimension de $\ker S$ comme $\mathbb{F}_p$ espace vectoriel est r.*

Ce résultat est important car il permet théoriquement de montrer que si f n'est pas irréductible, alors on peut trouver un polynôme non constant dans $\ker S$, et il donne pratiquement un critère de terminaison de l'algorithme.

Le calcul de l'endomorphisme S se fait via l'estimation des congruences suivantes :

$$x^{ip} - x^i \equiv \sum_{j=0}^{\deg f - 1} s_{ij} x^j \mod f,\ 0 \leq i \leq \deg f - 1$$

et donc si on représente un élément $t \in A$ par un vecteur colonne donnant ses coefficients dans la base $(1, x \dots x^{\deg f-1})$, la relation $St = 0$ permet de trouver les éléments du noyau par résolution d'un système linéaire. (Les indéterminées étant bien sûr les composantes de t)

2.3 Coût de l'algorithme

Venons-en maintenant au coût de l'algorithme de Berlekamp. Trois étapes importantes interviennent pour trouver un facteur non constant ou prouver l'irréductibilité.

Tout d'abord, rappelons les coûts des opérations « élémentaires » :

Proposition 2.3.1 *Soient $f, g \in \mathbb{Z}[X]$, avec g unitaire.*

La division euclidienne de f par g peut se calculer en $O(\deg g \times (\deg f - \deg g + 1))$ opérations élémentaires.

Le pgcd *de f et g se calcule en $O(\deg f \times \deg g)$ opérations élémentaires.*

Preuve :

► L'algorithme de division euclidienne est très simple : il consiste à retirer à f un polynôme de la forme $cX^i g$ afin de faire disparaître le terme de degré le plus haut de f. Ceci coûte $O(\deg g)$ opérations. On recommence autant de fois que nécessaire mais au plus $\deg f - \deg g + 1$ fois.

Pour le calcul du pgcd, on déroule l'algorithme d'Euclide : en supposant $\deg f \geqslant \deg g$, on effectue la division euclidienne $f = ug + r_0$ et on recommence avec g et r_0. D'où une complexité de l'ordre de :

$$\begin{aligned}
&C\left(\deg g(\deg f - \deg g + 1) + \deg r_0(\deg g - \deg r_0 + 1) + \sum \deg r_i(\deg r_{i-1} - \deg r_i + 1)\right)\\
=&C\left(\deg f \deg g - \deg g(\deg g - \deg r) - \sum \deg r_i(\deg r_{i-1} - \deg r_i) + \deg g + \sum \deg r_i\right)\\
\leqslant&C\left(\deg f \deg g + (\deg g)^2\right) \leqslant 2C \deg f \deg g.
\end{aligned}$$

◄

Remarquons que les opérations dans $\mathbb{F}_p$ sont faciles, d'une complexité au plus $(\log p)^2$, ce que l'on assimilera à une constante pour la suite †.

L'algorithme de Berlekamp résulte de la combinaison des trois étapes :

- le calcul de l'endomorphisme S
- la résolution du système linéaire $St = 0$
- les calculs des $\text{pgcd}(Q, t - \alpha),\ \alpha \in \mathbb{F}_p$.

Dans la suite, on notera n le degré du polynôme Q. La première étape consiste essentiellement en n divisions euclidiennes de polynômes dont le degré est majoré par np : elle se fait donc en $O(n \times (n(np - n)) = O(pn^3)$.

La deuxième est un pivot de Gauss sur un système de taille n, cela se fait donc en $O(n^3)$.

Enfin la troisième étape fait intervenir p divisions euclidiennes de polynômes de degré majoré par n, d'où un $O(p \times n^2)$.

Au total on a $O(p \times n^3)$ opérations pour trouver un facteur non constant ou pour prouver l'irréductibilité.

†. Voir plus loin (paragraphe 5.1) pour le choix de p, qui en pratique n'excède pas 100.

3 Lemme de Hensel

L'algorithme de Berlekamp nous permet d'obtenir la factorisation dans $\mathbb{Z}/p\mathbb{Z}$ (éventuellement l'irréductibilité) d'un polynôme à coefficients entiers. On cherche à remonter cette factorisation modulo $p^{2^k}, k \geq 1$. Le lemme de Hensel va exaucer ce souhait.

Lemme 3.1 *Soit P un polynôme à coefficients entiers unitaire, et soit :*

$$P = G_1 H_1 \mod p$$

une factorisation de P en produit d'éléments premiers entre eux. Notons U_1 et V_1 des polynômes unitaires vérifiant :

$$\deg U_1 < \deg H_1 \text{ et } \deg V_1 < \deg G_1,$$
$$U_1 G_1 + V_1 H_1 = 1 \mod p.$$

Alors pour tout $K = 2^k$ on a l'existence d'uniques G_K, H_K, U_K, V_K dans $\mathbb{Z}/p^K\mathbb{Z}[X]$ tels que :

$$G_K = G_{2K} \mod p^K, \text{ et de même pour } H, U, V,$$
$$G_K,\ H_K \text{ sont unitaires,}$$
$$P = G_K H_K \text{ dans } \mathbb{Z}/p^K\mathbb{Z},$$
$$U_K G_K + V_K H_K = 1 \text{ dans } \mathbb{Z}/p^K\mathbb{Z},$$
$$\deg U_K < \deg H_K,\ \deg V_K < \deg G_K.$$

<u>*Preuve*</u> :

▶ On va procéder par récurrence en supposant les polynômes H_K, G_K, V_K, U_K construits au rang K. Dès lors on pose le système d'équations :

$$\begin{cases} G_{2K} = G_K + p^K g_K \\ H_{2K} = H_K + p^K h_K \end{cases} \tag{1}$$

où $h_K, g_K \in \mathbb{Z}/p^K\mathbb{Z}[X]$ les inconnues. On réécrit l'égalité $P - G_K H_K = 0 \mod p^K$ en :

$$P - G_K H_K = p^K R_K \mod p^{2K}$$

En se rappelant qu'on veut $P = G_{2K} H_{2K} \mod p^{2k}$, le système (1) se réécrit en :

$$G_K h_K + H_K g_K = R_K \mod p^K$$

Il suffit donc de prendre :

$$\begin{cases} h_K = U_K R_K & \mod H_K, \\ g_K = V_K R_K & \mod G_K. \end{cases}$$

La construction de U_{2K} et V_{2K} se fait aussi de manière récursive. On remarque que U_K est l'inverse de G_K modulo H_K (et de même pour V_K et H_K). La question se ramène donc à trouver l'inverse de G_{2K} connaissant celui de G_K. Posons $U_{2K} = 2U_K - G_{2K}U_K^2 \mod H_{2K}$. On effectue les calculs suivant modulo p^{2K} et on a :

$$\begin{aligned} U_{2K}G_{2K} &= 2U_K G_{2K} - G_{2K}^2 U_K^2 && \mod H_{2K} \\ &= 1 - (U_K G_{2K} - 1)^2 && \mod H_{2K} \end{aligned}$$

or on a :

$$\begin{aligned} U_K G_{2K} &= U_K(G_K + p^K g_K) \\ &= 1 - H_K V_K + U_K p^K g_K \end{aligned}$$

ainsi

$$\begin{aligned}(U_K G_{2K} - 1)^2 &= (U_K p^K g_K - H_K V_K)^2 && \bmod H_{2K}\\ &= (U_K p^K g_K - (H_{2K} - p^K h_K) V_K)^2 && \bmod H_{2K}\\ &= 0 && \bmod H_{2K}\end{aligned}$$

car $H_K = H_{2K} - p^K h_K$ et tous les calculs se font modulo p^{2K}. C'est ce que l'on voulait démontrer. On obtient des formules analogues pour V_K. ◀

Remarquons que le lemme de Hensel est un peu plus général puisqu'il affirme que l'on peut remonter une factorisation de $\mathbb{F}_p[X]$ dans $\mathbb{Z}/p^k\mathbb{Z}$ pour tout $k \in \mathbb{N}^*$. Néanmoins, ce relèvement quadratique sera amplement suffisant et la preuve présentée a l'avantage de donner directement un algorithme de relèvement :

Algorithme 1 HENSEL

Entrées : $f, g, h, u, v \in \mathbb{Z}[X]$, $p, k \in \mathbb{N}$ *p premier,* $f = gh[p^k]$ *et* $ug + vh = 1[p^k]$.
Sorties : $\tilde{g}, \tilde{h}, \tilde{u}, \tilde{v} \in \mathbb{Z}[X]$, $f = \tilde{g}\tilde{h}[p^{2k}]$ *et* $\tilde{u}\tilde{g} + \tilde{v}\tilde{h} = 1[p^{2k}]$.

$r \leftarrow \frac{1}{p^k}(f - gh)$
$\tilde{h} \leftarrow h + p^k \times$ DIV_EUCLIDE(ur, h)
$\tilde{g} \leftarrow g + p^k \times$ DIV_EUCLIDE(vr, g)
$\tilde{u} \leftarrow$ DIV_EUCLIDE$(2u - u^2\tilde{g}, \tilde{h})$
$\tilde{v} \leftarrow$ DIV_EUCLIDE$(2v - v^2\tilde{h}, \tilde{g})$
Rendre $\tilde{g}, \tilde{h}, \tilde{u}, \tilde{v}$

Proposition 3.2 *L'algorithme de relèvement présenté ci-dessus est polynomial en le logarithme des coefficients de f, en le degré de f, en p et en k.*

Preuve :
▶ En effet, tous les coefficients des polynômes sont majorés par $\max(|a_i|, p^{2k})$ où les a_i sont les coefficients de f. De plus, tous les degrés de ces polynômes sont majorés par n.

Ensuite, les seules opérations utilisées sont la multiplication, l'addition et la division euclidienne des polynômes, qui sont bien de coût polynomial en les données.

En supposant que $p^k > \max|a_i|$, ce qui correspond au cadre dans lequel on va utiliser l'algorithme, on peut estimer le coût de cet algorithme à $O(k^2 \times n^2)$ où la présence du k^2 s'explique par le fait que les opérations arithmétiques de base ne peuvent plus être considérées comme temps constant. ◀

Remarquons que l'on peut relever une factorisation de $\mathbb{F}_p[X]$ dans $\mathbb{Z}/p^{2^k}\mathbb{Z}$ en temps polynomial si 2^k est polynomial en les données : en effet toutes les remontées le seront et le nombre de remontées aussi.

4 L'algorithme LLL

Bien que de nature apparemment géométrique, l'algorithme LLL, du nom de ses inventeurs A. Lenstra, H. Lenstra et L. Lovász, a été initialement conçu pour la factorisation des polynômes. Étant donné un réseau de $\mathbb{Q}^n$,‡ son but est de trouver une base de vecteurs presque orthogonaux dont les normes sont graduellement petites.

‡. On peut se placer sur $\mathbb{R}$ mais cela n'a que peu de sens d'un point de vue informatique.

Définition 4.0.2 (Réseaux) *Un réseau de $\mathbb{Q}^n$ est un sous $\mathbb{Z}-$module de $\mathbb{Q}^n$ de type fini. La dimension d'un réseau est le cardinal d'une partie génératrice minimale.*

Généralement, un réseau est présenté comme un sous-groupe discret [◇] de $\mathbb{R}^n$. Un résultat classique montre en fait que tout sous-groupe discret de $\mathbb{R}^n$ est isomorphe à $\mathbb{Z}^d$ avec $d \leqslant n$. La dimension est alors d et les deux visions sont essentiellement les mêmes. Notons qu'une partie génératrice minimale peut être de cardinal strictement plus grand que n.

Ceci dit, dans la suite, on confondra parfois une famille libre [⋆] et le réseau qu'elle engendre et la dimension sera simplement le cardinal de la famille libre.

Définition 4.0.3 *Soit $m \leqslant n$. On appelle déterminant du réseau engendré par une famille libre $A = (a_1, \ldots, a_m)$ la quantité $\Delta = \det({}^tAA)$*

On vérifie facilement qu'il ne dépend pas de la base choisie.

4.1 Orthogonalisation de Gram–Schmidt

Dans toute la suite, nous noterons $\|.\|$ la norme euclidienne d'un vecteur. On identifiera le polynôme $f = \sum_{i=0}^{n} a_i X^i$ avec le vecteur formé de ces coefficients $(a_0, \ldots, a_n)$ et $\|f\| = \|(a_0, \ldots, a_n)\|$.

Rappelons brièvement le procédé d'orthogonalisation de Gram–Schmidt, afin de fixer des notations. Étant donné une famille libre de vecteurs $(b_1, \ldots, b_m)$ avec $m \leqslant n$, on veut trouver une famille orthogonale $(b_1^*, \ldots, b_m^*)$ vérifiant $b_k^* \in \underset{i \in [\![1,k]\!]}{\text{vect}}(b_i)$ pour tout k.

Proposition 4.1.1 (Orthogonalisation de Gram–Schmidt) *Une telle famille existe et s'obtient par l'algorithme suivant. On pose $b_1^* \leftarrow b_1$ et pour i allant de 2 à m on effectue :*

$$\mu_{ij} \leftarrow \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)} \qquad (j = 1, \ldots, i-1)$$
$$b_i^* \leftarrow b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*.$$

Il s'agit simplement de projeter le vecteur b_i sur la famille orthogonale déjà obtenue $(b_1^*, \ldots, b_{i-1}^*)$.

Cette procédure d'orthogonalisation prend un temps $O(n \times m)$ opérations élémentaires sur les rationnels. Si l'on note s le nombre de bits maximal qu'il faut pour stocker un numérateur ou un dénominateur d'une des données, la nouvelle base aura des numérateurs et dénominateurs dont la taille sera majorée en $O(m \times s)$, ce qui assure la complexité polynomiale de l'algorithme.

◇. Un sous-groupe de $\mathbb{R}^n$ est discret si son intersection avec tout compact est finie.

⋆. On ne précise pas si la notion de liberté est dans $\mathbb{Q}^n$ ou en tant que $\mathbb{Z}-$module car les deux notions coïncident.

4.2 Bases faiblement réduites

Le but de l'algorithme LLL est de donner une base d'un réseau presque orthogonale, en fournissant des petits vecteurs.

Pour cela, on introduit des notions de bases réduites.

Définition 4.2.1 (Réduction faible) *Une base $B = (b_1, \ldots, b_m)$ sera dite faiblement réduite si en l'orthogonalisant, on obtient la condition pour tout $1 \leqslant j < i \leqslant m$:*

$$|\mu_{ij}| \leqslant \frac{1}{2}.$$

Cette condition est très naturelle car on ne peut faire que des combinaisons linéaires entières lors des changements de bases : on ne peut donc se rapprocher de «l'idéal» orthogonal qu'à $\frac{1}{2}$ près.

Proposition 4.2.2 *Toute base peut être faiblement réduite.*

Preuve :

► Soit (i_0, j_0) le plus grand couple pour l'ordre lexicographique qui ne vérifie pas $|\mu_{i_0 j_0}| \leqslant \frac{1}{2}$ (parmi les couples (i, j), $i > j$). Soit $c = \lfloor \mu_{i_0 j_0} \rceil$ l'entier le plus proche de $\mu_{i_0 j_0}$. On transforme B en la base $\overline{B}$ par l'opération :

$$\overline{b}_i = \begin{cases} b_i & si\ i \neq i_0 \\ b_{i_0} - c b_{j_0} & sinon. \end{cases}$$

Cette transformation ne change pas les vecteurs de l'orthogonalisée ($B^* = \overline{B}^*$). Seuls les coefficients μ_{ij} ont changé. En écrivant $b_{i_0} - c b_{j_0} = b^*_{i_0} + \sum_{j<i_0} \mu_{i_0 j} b^*_j - c b^*_{j_0} - c \sum_{j<j_0} \mu_{j_0 j} b^*_j$, on trouve :

$$\overline{\mu}_{ij} = \begin{cases} \mu_{ij} & si\ i \neq i_0\ ou\ si\ j > j_0 \\ \mu_{i_0 j} - c\mu_{j_0 j} & sinon. \end{cases}$$

En particulier les couples plus grands que (i_0, j_0) ne sont pas modifiés et $|\overline{\mu}_{i_0 j_0}| = |\mu_{i_0 j_0} - c| \leqslant \frac{1}{2}$.

En réitérant cette opération au plus $\binom{m}{2}$ fois, on peut donc faiblement–réduire la base B en $O(m^2 \times n)$ opérations. ◄

4.3 Bases réduites

Définition 4.3.1 *Une base B d'un réseau sera dite réduite si elle est faiblement réduite et vérifie de plus pour $1 \leqslant i < m$*

$$\|b^*_{i+1} + \mu_{i+1,i} b^*_i\|^2 \geqslant \frac{3}{4} \|b^*_i\|^2. \tag{2}$$

Rappelons qu'une réduction faible ne modifie pas l'orthogonalisée et donc peut intervenir à tout moment dans un algorithme de réduction, sans modifier ce qui a déjà été fait.

Le but de ce qui va suivre est de montrer le résultat suivant :

Théorème 4.3.2 *Toute base peut être réduite. De plus, le temps de la réduction est polynomiale en la taille des données.*

Pour cela, nous allons montrer quelques propriétés des bases réduites. Tout d'abord, introduisons des notations :

Notations 4.3.3 *Soit un réseau engendré par une $\mathbb{Z}-$base $B = (b_1, \dots, b_m)$.*

Pour $1 \leqslant i \leqslant m$, on note Δ_i (ou $\Delta_i(B)$) le déterminant du réseau engendré par $B_i = (b_1, \dots, b_i)$.

On note de plus $V(B) = \prod_{i=1}^{m} \Delta_i(B)$.

On remarque facilement, du fait que la matrice de changement de base vers l'orthogonalisée est triangulaire supérieure de diagonale $(1, \dots, 1)$, que $\Delta_i = \prod_{j=1}^{i} \|b_j^*\|^2$.

Ainsi, une réduction faible donne une nouvelle base B' on a $V(B') = V(B)$.

Proposition 4.3.4 *On a l'inégalité :*

$$V(B) \leqslant \left(\max_i \|b_i\|\right)^{m(m+1)}.$$

Preuve :

► En écrivant $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{ij} b_j^*$ on obtient $\|b_i\| \geqslant \|b_i^*\|$ pour tout i. On a alors :

$$\begin{aligned} V(B) = \prod_{i=1}^{m} \Delta_i(B) = \prod_{i=1}^{m} \prod_{j=1}^{i} \|b_j^*\|^2 &= \prod_{i=1}^{m} \|b_i^*\|^{2(m-i+1)} \\ &\leqslant \prod_{i=1}^{m} \|b_i\|^{2(m-i+1)} \\ &\leqslant \left(\max_i \|b_i\|\right)^{m(m+1)}. \end{aligned}$$

◄

Voici enfin la proposition clé dans l'algorithme de réduction.

Proposition 4.3.5 *Si la base B du réseau est faiblement réduite et si les vecteurs b_i^* et b_{i+1}^* ne vérifient pas l'inégalité (2) alors la base $C = (b_1, \dots, b_{i+1}, b_i, \dots, b_m)$ vérifie :*

$$V(C) < \frac{3}{4} V(B).$$

Preuve :

► Il s'agit de faire le lien entre C^* et B^*. Il est clair que pour $j \neq i$ on a :

$$\Delta_j(B) = \Delta_j(C)$$

puisque les réseaux sont les mêmes. Ainsi $\frac{V(C)}{V(B)} = \frac{\Delta_i(C)}{\Delta_i(B)} = \frac{\|c_i^*\|^2}{\|b_i^*\|^2}$. Il faut alors relier $\|c_i^*\|$ à $\|b_i^*\|$.

On a $c_i = b_{i+1} = b_{i+1}^* + \sum_{j=1}^{i} \mu_{i+1,j} b_j^* = b_{i+1}^* + \mu_{i+1,i} b_i^* + \sum_{j=1}^{i-1} \nu_{ij} c_j^*$, où on a noté ν_{ij} les coefficients d'orthogonalisation de C. Ceci vient du fait que pour $j \leqslant i-1$, $c_j^* = b_j^*$ est orthogonal à $b_{i+1}^* + \mu_{i+1,i} b_i^*$. On a alors :

$$\begin{aligned} c_i^* &= c_i - \sum_{j=1}^{i-1} \nu_{ij} c_j^* \\ &= b_{i+1}^* + \mu_{i+1,i} b_i^* \end{aligned}$$

et par hypothèse $\|b_{i+1}^* + \mu_{i+1,i} b_i^*\|^2 < \frac{3}{4} \|b_i^*\|^2$. ◄

Cette proposition fournit alors une idée raisonnable d'algorithme pour réduire une base (au sens fort) : on trouve le plus petit indice qui ne vérifie pas la condition de réduction forte (2), on échange ces deux vecteurs et on réduit faiblement la base. On recommence alors récursivement.

C'est le facteur de réduction géométrique dans la proposition précédente qui fait marcher l'algorithme en temps polynomial . Notons que le point important est que $\frac{3}{4} < 1$ et que d'autres facteurs peuvent tout aussi bien marcher. ††

Algorithme 2 LLL

Entrée : $B \in \mathbb{Q}^{n\times m}$
Sortie : $\overline{B} \in \mathbb{Q}^{n\times m}$ *base réduite.*

$\overline{B} \leftarrow$REDUIT–FAIBLE (B)
Tant que $\overline{B}$ n'est pas réduite **Faire**
  $i \leftarrow \min_j \left(\|b_j^* + \mu_{j+1,j} b_{j+1}^*\|^2 < \frac{3}{4}\|b_j^*\|^2\right)$
  échanger $b_i \leftrightarrow b_{i+1}$
  $\overline{B} \leftarrow$REDUIT–FAIBLE $(\overline{B})$
Fin Tant que
Rendre $\overline{B}$

Proposition 4.3.6 (Terminaison et complexité) *La boucle « Tant que » ne s'exécute qu'un nombre de fois polynomial en la taille des données.*

Preuve :
▶ Soit d le pgcd des dénominateurs. On a $C = dB$ est à valeurs entières et donc $|V(\overline{C})| \geqslant 1$ à toutes les étapes de l'algorithme lancé sur C. La boucle « Tant que » ne peut donc s'exécuter plus que $\log_{\frac{3}{4}}(V(C))$ fois. Or $\Delta_i(C) = d^{2i}\Delta_i(B)$ et donc, avec la proposition 4.3.4 :

$$V(C) = d^{m(m+1)}V(B) \leqslant \left(d \max_i \|b_i\|\right)^{m(m+1)}.$$

Si s est le nombre maximal de bits nécessités par un numérateur ou dénominateu,r on a un nombre d'itérations en $O(m^2(s + \log d))$. ◀

Ceci permet d'énoncer le théorème suivant :

Théorème 4.3.7 *L'algorithme* LLL *s'exécute en temps polynomial en les données.*

Preuve :
▶ L'étape de réduction faible (qui consiste aussi à mettre à jour les μ_{ij}) ne nécessite pas d'être effectuée sur toute la matrice : en effet, les vecteurs $(b_1, \ldots, b_{i-1})$ ne sont pas changés et les coefficients μ correspondant non plus. Par ailleurs, on ne maîtrise rien de ce qui se passe aux rangs supérieurs à $i + 1$ et il ne sert à rien de réduire faiblement au delà de $i + 1$. Tout ceci permet de voir que l'étape de réduction faible ne peut coûter que $O(n^2)$ si elle est bien programmée.

On a donc une complexité totale de LLL en $O[m^2 \times n^2 \times (s + \log d)]$ qui est majorée au pire des cas (tous les dénominateurs sont grands et premiers entre eux ‡‡) par $O(m^4 \times n^2 \times s)$.

Ceci n'achève pas la preuve car on ne sait pas si les rationnels dans les calculs intermédiaires « n'explosent » pas en taille mémoire. C'est l'objet du lemme suivant. ◀

††. Le choix de $\frac{3}{4}$ est surtout motivé par le fait qu'il s'agit du facteur utilisé dans l'article original [1].
‡‡. On a alors $\log(d) = O(m^2 \times s)$.

Lemme 4.3.8 *On suppose que le réseau est à valeurs dans $\mathbb{Z}^n$. Notons M=$\max\left(\max_i \|b_i\|, 1\right)$ et $s = \log M$. Alors les entiers intervenants dans les calculs ont leur logarithme borné par :*

$$O\left(n \log \max_i \|b_i\|\right) = O(ns).$$

Le cas des réseaux à valeurs dans $\mathbb{Q}$ s'en déduit mais nous nous contenterons de ce lemme car il correspond au cadre dans lequel on utilisera LLL.
Preuve : (idée)
► Un calcul facile montre que tout au long de l'algorithme $\|b_i^*\| \leqslant M$. Ensuite, si l'on note k le plus petit entier qui fait défaut à la condition de réduction forte, alors on a encore, au moment de l'échange de b_k et b_{k+1} et avant l'étape de réduction faible :

$$|\mu_{ij}| \leqslant \frac{1}{2},\ \forall 1 \leqslant j < i \leqslant k.$$

Ainsi $\|b_i\|^2 \leqslant nM^2$ pour $i \leqslant k$. De plus, pour $i > k+1$ les vecteurs ne sont pas modifiés et par une induction facile, on montre que :

$$\forall i \neq k+1,\ \|b_i\|^2 \leqslant nM^2.$$

Il nous reste à traiter le cas $k+1$. Pour cela, on montre la propriété suivante par récurrence :

$$|\mu_{k+1,j}| \leqslant 2^{n-(k+1)}\left(M^{n-1}\right),\ j \leqslant k.$$

Au début de l'algorithme, on a pour tout couple (i,j) avec[◇◇] $j \leqslant n-1$:

$$\mu_{ij}^2 \leqslant \frac{\|b_i\|\|b_j^*\|}{\|b_j^*\|^2} \leqslant \frac{M^2}{\|b_j^*\|^2} \leqslant \frac{\Delta_{j-1}M^2}{\Delta_j} \leqslant \Delta_{j-1}M^2 \leqslant M^{2j} \leqslant M^{2(n-1)}.$$

On prouve alors l'hérédité. On note ν les nouveaux coefficients après l'échange de b_k et b_{k+1}. On a alors par un calcul simple, pour $j < k$:

$$\begin{aligned}
|\nu_{kj}| = |\mu_{k+1,j} - c\mu_{kj}| &< |\mu_{k+1,j}| + \frac{c}{2} \\
&< |\mu_{k+1,j}| + |\mu_{k+1,k}| \\
&< 2^{n-(k+1)}\left(M^{n-1}\right) + 2^{n-(k+1))}\left(M^{n-1}\right) \\
&< 2^{n-k}\left(M^{n-1}\right)
\end{aligned}$$

car $|c| \leqslant 2|\mu_{k+1,k}|$. Ceci est bien l'inégalité attendue car à la fin de l'étape, k devient $k-1$. Ceci termine la récurrence.
Dès lors on peut majorer b_{k+1} :

$$\begin{aligned}
\|b_{k+1}\|^2 &\leqslant \|b_{k+1}^*\|^2 + \sum_{j=1}^{k} |\mu_{k+1,j}|^2 \|b_j^*\|^2 \\
&\leqslant M^2 + k(2^{2n}M^{2(n-1)}M^2) \leqslant n2^{2n}M^{2n}.
\end{aligned}$$

Ainsi on a bien $\log \|b_i\| = O(n \log M)$. La formule des μ_{ij} rappelée au début de la section 4.1 assure que la taille de leur dénominateur[⋆⋆] est bornée par $\log M$ et celle de leur numérateur bornée par $O(n \log M + \log M) = O(n \log M)$. Ceci achève la démonstration du lemme. ◄

◇◇. On prend $j \leqslant n-1$ car on l'applique à $j \leqslant k < k+1 \leqslant n$.
⋆⋆. *i.e.* leur logarithme.

Dans notre application à la factorisation de polynômes, le réseau initial sera à valeurs dans $\mathbb{Z}^n$ avec $m = n$ et on pourra retenir une complexité en $O(n^4 \times s)$ opérations élémentaires. Comme de plus on travaille sur des grands entiers, il faut ajouter un facteur $(ns)^2$ pour tenir compte du temps des opérations arithmétiques élémentaires et on a un temps de l'ordre de $O(n^6 \times s^3)$.

Finissons la présentation de l'algorithme LLL par des propriétés intéressantes de la base réduite.

Proposition 4.3.9 *Si B est une base réduite de dimension n, on a :*

1. $\Delta_n(B) \leqslant \prod_{i=1}^n \|b_i\|^2 \leqslant 2^{\frac{n(n-1)}{2}} \Delta_n(B)$.
2. $\|b_1\|^2 \leqslant 2^{\frac{n(n-1)}{2}} \Delta_n(B)^{\frac{2}{n}}$.
3. $\|b_j\|^2 \leqslant 2^{i-1} \|b_i^*\|^2$ *pour* $1 \leqslant j \leqslant i \leqslant n$.
4. *Pour* $x \in \operatorname{vect} B$, $x \neq 0$, $\|b_1\|^2 \leqslant 2^{n-1}\|x\|^2$.

Preuve :

► La première inégalité du premier point est l'inégalité d'Hadamard et ne nécessite aucune propriété de réduction de la base B. Nous ne donnons une preuve que des deux derniers points car ce sont les seuls qui vont nous servir dans la suite. Toutes les références sur LLL les détaillent.

De la définition (2) des bases réduites, on tire $\|b_{i+1}^*\|^2 \geqslant \left(\frac{3}{4} - \mu_{i+1,i}\right) \|b_i\|^2 \geqslant \frac{1}{2}\|b_i^*\|^2$. Par une récurrence facile, on en tire pour tout $1 \leqslant j \leqslant i \leqslant n$:

$$\|b_j^*\|^2 \leqslant 2^{i-j}\|b_i^*\|^2. \tag{3}$$

Il suffit alors de décomposer b_i sur $(b_1^*, \ldots, b_i^*)$ et de calculer sa norme :

$$\begin{aligned}
\|b_i\|^2 &= \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 \|b_j^*\|^2 \\
&\leqslant \|b_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j} \|b_i^*\|^2 \\
&\leqslant 2^{i-1}\|b_i^*\|^2.
\end{aligned}$$

De cette inégalité et de 3, on tire $\|b_j\|^2 \leqslant 2^{j-1}\|b_j^*\|^2 \leqslant 2^{i-1}\|b_i^*\|^2$.

Il nous reste à établir le dernier point. Pour cela, on décompose x sur la base $(b_1, \ldots, b_n)$:

$$x = \sum_{i=1}^{j} n_i b_i = \sum_{i=1}^{j} r_i b_i^*$$

avec $r_j = n_j \neq 0$ et les n_i entiers. On en déduit $\|x\|^2 \geqslant n_i^2 \|b_j^*\|^2 \geqslant \|b_j^*\|^2$. En combinant avec le point précédent on a :

$$\begin{aligned}
\|x\|^2 &\geqslant 2^{1-i}\|b_1\|^2 \\
&\geqslant 2^{1-n}\|b_1\|^2.
\end{aligned}$$

◄

Cette dernière inégalité nous montre qu'à un facteur 2^{n-1} près, b_1 est un petit vecteur de réseau. Ceci est d'autant plus intéressant que trouver un plus petit vecteur d'un réseau ayant une base donnée est un problème soupçonné comme étant **NP**−complet.

5 Factorisation dans $\mathbb{Z}[X]$

5.1 Premières étapes

Tout d'abord, notons que le problème de la factorisation sur $\mathbb{Q}[X]$ est le même, quitte à multiplier par le dénominateur commun à tous les coefficients. Notons de plus que l'on peut supposer que le polynôme de départ est sans facteurs carrés car sinon on peut déjà factoriser par $\text{pgcd}(f, f')$ qui est non trivial. Finalement on peut supposer que le polynôme est unitaire. En effet on peut réécrire :

$$f(X) = \sum_{i=0}^{n} a_i X^i = \frac{1}{a_n^{n-1}} \left((a_n X)^n + \sum_{i=0}^{n-1} a_i a_n^{n-1-i} (a_n X)^i \right) = \frac{1}{a_n^{n-1}} \tilde{f}(a_n X).$$

Il suffit de factoriser $\tilde{f}$ qui est unitaire, pour pouvoir ensuite factoriser f.

Dans la suite, on supposera $f \in \mathbb{Z}[X]$ unitaire et sans facteurs carrés.
Il s'agit d'utiliser les différents algorithmes présentés ci-dessus pour aboutir à une factorisation de f. La première étape consiste à factoriser f dans $\mathbb{F}_p$ pour p choisi de telle façon que f n'ait pas de facteurs carrés dans ce corps. Pour cela, on calcule le résultant de f et f' (qui est non nul [†††]) et on choisit le plus petit p qui ne divise pas ce résultant. Remarquons que p n'est pas trop grand :

Proposition 5.1.1 *Pour n assez grand, $p \leqslant 4n \log(\max |a_i|) + 4n \log n$, où $f = \sum_{i=0}^{n} a_i X^i$.*

Preuve :
▶ En effet, on a par l'inégalité d'Hadamard :

$$|\text{res}(f, f')| \leqslant \|f\|^{n-1} \|f'\|^n \leqslant (\sqrt{n} \max |a_i|)^{n-1} (\sqrt{n} n \max |a_i|)^n \leqslant n^{2n} (\max |a_i|)^{2n}.$$

Calculons le nombre maximal de facteurs premiers distincts qui peuvent diviser ce résultant. Pour cela, on utilise le théorème des nombres premiers qui affirme que :

$$\sum_{p \leqslant m} \log p \sim m.$$

Ainsi, pour m assez grand, $\prod_{p \leqslant m} p \geqslant e^{\frac{m}{2}}$ et on a donc l'implication

$$m \geqslant 4n \log n + 4n \log(\max |a_i|) \Longrightarrow \prod_{p \leqslant m} p > |\text{res}(f, f')|.$$

Ainsi, $\text{res}(f, f')$ ne peut admettre plus de $4n \log n + 4n \log(\max |a_i|)$ facteurs premiers distincts. On peut donc trouver un $p \nmid \text{res}(f, f')$ dont la taille est polynomiale en les coefficients $(\log(\max |a_i|))$ et en le degré de f. En pratique, il est rare que p soit bien plus grand que 100. ◀

Une fois une factorisation effectuée de $f = gh$ dans $\mathbb{F}_p$, l'idée est de relever cette factorisation en $f = \hat{g}\hat{h}$ dans $\mathbb{Z}/p^k\mathbb{Z}$ grâce au lemme de Hensel pour un k bien choisi. On note $l = \deg g$ et on utilise ensuite LLL pour le réseau engendré par :

$$\{p^k X^i, 0 \leqslant i < l\} \cup \{\hat{g} X^i, 0 \leqslant i < n - l\}.$$

Dans la suite de cette section, on va s'attacher à montrer que le plus petit vecteur d'une base réduite de ce réseau a soit un facteur non trivial en commun avec f soit f est irréductible.

†††. $\text{res}(f, f') \neq 0 \Leftrightarrow f$ est sans facteurs carrés

5.2 Propriétés du relèvement

Voyons tout d'abord un lemme tout à fait général qui commence à faire apparaître l'intérêt des petits vecteurs dans la recherche de facteurs non triviaux.

Lemme 5.2.1 *Soient $f, h \in \mathbb{Z}[X]$ de degrés respectifs n et l. Supposons par ailleurs qu'il existe $g \in \mathbb{Z}[X]$ unitaire et non constant qui divise f et h modulo $m \in \mathbb{N}^*$ avec $\|f\|^l \|h\|^n < m$. Alors,* $\gcd(f, h)$ *n'est pas constant.*

Preuve :
► On sait qu'il existe des polynômes $u, v \in \mathbb{Z}[X]$ tels que $uf + vh = \mathrm{res}(f, h)$. En prenant cette relation modulo m, on a l'existence d'un polynôme $w \in \mathbb{Z}[X]$ tel que $gw = \mathrm{res}(f, h)[m]$. Comme g est unitaire et non constant, on a $w = 0[m]$ et donc m divise $\mathrm{res}(f, h)$.

D'un autre côté, l'inégalité d'Hadamard donne $|\mathrm{res}(f, h)| \leqslant \|f\|^l \|h\|^n < m$. Ainsi, $\mathrm{res}(f, h) = 0$ et donc $\gcd(f, h)$ n'est pas constant. ◄

L'utilisation de ce lemme dans notre cadre est presque immédiate. Soit $g \in \mathbb{Z}[X]$ un polynôme unitaire irréductible dans $\mathbb{F}_p[X]$ facteur simple de f modulo p et facteur de f modulo p^k, via le lemme de Hensel. On note $l > 0$ son degré. On pose $m = p^k$ et on considère, comme annoncé, le réseau formé des polynômes de $\mathbb{Z}_{n-1}[X]$ qui modulo p^k sont multiples de g :

$$\{p^k X^i, 0 \leqslant i < l\} \cup \{gX^i, 0 \leqslant i < n - l\}. \tag{4}$$

Dans ces conditions, le lemme précédent permet d'affirmer :

Proposition 5.2.2 *Soit b un vecteur non nul de ce réseau vérifiant $\|b\|^n \|f\|^{n-1} < p^k$. Alors,* $\gcd(f, b)$ *est un facteur non trivial de f dans $\mathbb{Z}[X]$.*

Preuve :
► C'est une conséquence triviale du lemme : il suffit de remarquer que b est bien divisible par g modulo p^k, que g est unitaire et non constant et que $\deg b \leqslant n - 1$. ◄

Cette proposition relativement élémentaire suffirait à construire un algorithme en temps polynomial. Voyons néanmoins une version plus avancée qui est en fait la propriété énoncée dans l'article historique de LLL. Même si elle ne permet pas de faire baisser la complexité au pire de l'algorithme, nous verrons que la complexité moyenne en est améliorée. Avant d'énoncer cette propriété, nous avons besoin d'un lemme sur les réseaux de $\mathbb{Z}^n$:

Lemme 5.2.3 *Soit Λ un réseau de $\mathbb{R}^n$ et $R \subset \Lambda$ un sous-réseau. Soit $(a_1, \ldots, a_n)$ une base de Λ. Alors on peut trouver une base $(b_1, \ldots, b_n)$ de R de la forme :*

$$\begin{array}{lllll} a_1 &= \nu_{11} b_1 & & & \\ a_2 &= \nu_{21} b_1 &+ \nu_{22} b_2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ a_n &= \nu_{n1} b_1 &+ \quad \ldots \quad &+ \nu_{nn} b_n. \end{array}$$

Preuve :
► Notons $B_k = \mathrm{vect}_{\mathbb{Z}}\{b_1, \ldots, b_k\}$.

Remarquons tout d'abord que si $x, y \in B_k$ alors on peut trouver $z \in B_{k-1}$. En effet, on écrit $x = x' + \alpha b_k$ et $y = y' + \beta b_k$ avec $x', y' \in B_{k-1}$, puis $d = \gcd(\alpha, \beta) = u\alpha + v\beta$. Alors, $z = x - \frac{\alpha}{d}(ux + vy)$ convient.

Par une récurrence simple, on en déduit que pour tout $1 \leqslant k \leqslant n$ on a $B_k \cap R \neq \emptyset$. Choisissons donc des vecteurs de R de la forme :

$$a_i = \nu_{i1} b_1 + \ldots \nu_{ii} b_i$$

où les ν_{ik} sont entiers et $|\nu_{ii}| > 0$ est minimal. On va montrer par l'absurde que $(a_1, \ldots, a_n)$ convient. Soit donc $c \in R \backslash \operatorname{vect}_{\mathbb{Z}}\{a_1, \ldots, a_n\}$ tel que $c = t_1 b_1 + \ldots + t_k b_k$ pour k minimal puis pour $|t_k|$ minimal. Alors, comme $\nu_{kk} \neq 0$, on peut trouver $s \in \mathbb{N}$ tel que :

$$|t_k - s\nu_{kk}| < |\nu_{kk}|.$$

Soit alors $d = c - sa_k \in R \backslash \operatorname{vect}_{\mathbb{Z}}\{a_1, \ldots, a_n\}$. Par minimalité de k, d a une composante non nulle sur b_k mais celle-ci $(t_k - s\nu kk)$ est strictement plus petite en valeur absolue que ν_{kk}, ce qui contredit la construction de a_k. ◄

Corollaire 5.2.4 *Tout réseau R de $\mathbb{Z}^n$ a une base échelonnée $(b_1, \ldots, b_k)$ c'est-à-dire vérifiant :*

$$b_i \in \operatorname{vect}_{\mathbb{Z}}\{e_1, \ldots, e_{i_j}\} \text{ avec } 1 \leqslant i_1 < \ldots < i_k \leqslant n.$$

Preuve :
► Il suffit d'appliquer le lemme au réseau Λ engendré par $\{e_{\deg f},\ f \in R\}$ où $(e_1, \ldots, e_n)$ est la base canonique de $\mathbb{Z}^n$. ◄

Venons-en maintenant à la proposition principale, qui améliore la proposition 5.2.2. Rappelons que l'on considère le réseau :

$$\{p^k X^i, 0 \leqslant i < l\} \cup \{gX^i, 0 \leqslant i < n - l\}$$

où g est un facteur irréductible de degré l qui divise f modulo p^k. Il est clair que f est divisible par un unique polynôme g_1, irréductible unitaire, multiple de g modulo p^k.

Proposition 5.2.5 *Soit b un vecteur non nul de ce réseau vérifiant $\|b\|^n \|f\|^{n-1} < p^{kl}$. Alors, b est divisible par g_1 dans $\mathbb{Z}[X]$.*

Preuve :
► Soit $h = \operatorname{pgcd}(f, b) \in \mathbb{Z}[X]$. Il est clair qu'il suffit de montrer que g divise h dans $\mathbb{F}_p[X]$. Nous raisonnons par l'absurde. Comme g est irréductible dans $\mathbb{F}_p[X]$ on a l'existence de polynômes λ, μ et $\nu \in \mathbb{Z}[X]$ tels que :

$$\lambda g + \mu h = 1 - p\nu.$$

Remarquons qu'en multipliant par $1 + p\nu + \ldots + p^{k-1}\nu^{k-1}$ on a :

$$\lambda' g + \mu' h \equiv 1[p^k]. \tag{5}$$

Du fait que g ne divise pas h, on en déduit aussi que g divise f/h dans $\mathbb{F}_p[X]$ et donc

$$\deg g \leqslant \deg f - \deg h. \tag{6}$$

Considérons le réseau engendré par :

$$R = \{\alpha f + \beta b,\ \deg \alpha < \deg b - \deg h,\ \deg \beta < \deg f - \deg h\}$$

et notons R' son projeté sur le réseau :

$$\mathbb{Z}X^{\deg h} + \ldots + \mathbb{Z}X^{\deg f + \deg b - \deg h - 1}.$$

Soit $c \in R$. Comme g divise f et b modulo p^k, g divise c modulo p^k. Or on a aussi h qui divise c dans $\mathbb{Z}[X]$. En multipliant (5) par c/h et en utilisant le fait que g divise c modulo p^k on a :

$$c/h \equiv 0[p^k, g].$$

Si $\deg c < \deg g + \deg h$ alors $c \in p^k\mathbb{Z}[X]$. Ceci permet de minorer le déterminant de R'. En effet, par le corollaire 5.2.4 on prend une base de R' triangulaire *i.e.* une base de polynômes dont les degrés s'échelonnent de $\deg h$ à $\deg f + \deg b - \deg h$. Or, par (6) $\deg f + \deg b - \deg h \geqslant \deg g + \deg b \geqslant \deg g + \deg h$. Ainsi, cette base contient au moins $\deg g$ vecteurs v vérifiant $\deg v < \deg g + \deg h$ et donc à coefficients dans $p^k\mathbb{Z}$. D'où en effectuant le produit des coefficients dominants des vecteurs de cette base :

$$\sqrt{\Delta R'} \geqslant (p^k)^{\deg g} = p^{kl}.$$

Ceci contredit l'inégalité d'Hadamard pour R'. En effet, montrons que les projections sur R' des vecteurs :

$$\{X^i f,\ 0 \leqslant i < \deg b - \deg h\} \cup \{X^j b,\ 0 \leqslant j < \deg f - \deg h\} \tag{7}$$

sont linéairement indépendants. En effet, si $\alpha f + \beta b$ se projette sur 0 dans R' alors $\deg(\alpha f + \beta b) < \deg h$ mais par ailleurs, h divise $\alpha f + \beta b$ dans $\mathbb{Q}[X]$. D'où $\alpha f + \beta b = 0$. On peut donc appliquer l'inégalité d'Hadamard à R' pour la base sont les projetés ‡‡‡ des vecteurs de (7) :

$$\sqrt{\Delta R'}| \leqslant \|f\|^{\deg b - \deg h}\|b\|^{\deg f - \deg h} \leqslant \|f\|^{n-1}\|b\|^n < p^{kl}. \qquad \blacktriangleleft$$

5.3 L'algorithme de factorisation

Voici maintenant le théorème principal qui permet de mettre en place un algorithme polynomial de la factorisation de polynômes :

Théorème 5.3.1 (LLL**)** *Choisissons k suffisamment grand pour que :*

$$\|f\|^{2n-1} n^{\frac{n}{2}} \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}^n 2^{\frac{n(n-1)}{2}} < p^{kl}. \tag{8}$$

Soit $(b_1, \ldots, b_n)$ une base LLL*–réduite de (4). Alors :*

- *soit* $\mathrm{pgcd}(b_1, f)$ *est un facteur non trivial de f,*
- *soit f est irréductible dans $\mathbb{Z}[X]$.*

Preuve :

► Supposons f réductible et montons que b_1 vérifie la condition de la proposition 5.2.5. Le facteur g_1, irréductible, unitaire et non trivial de f vérifie $g_1 \equiv 0[p^k, g]$ et donc g_1 est dans le réseau (4). Par la proposition 4.3.9 on a :

$$\|g_1\| \geqslant 2^{\frac{1-n}{2}}\|b_1\|.$$

Ainsi, $\|b_1\|^n\|f\|^{n-1} \leqslant \|f\|^{n-1}\|g_1\|^n 2^{\frac{n(n-1)}{2}}$. Il reste à relier $\|f\|$ et $\|g_1\|$. Pour cela, on utilise un résultat dû à Mignotte ◇◇◇ :

$$\|g_1\| \leqslant \sqrt{n}\binom{\deg g_1}{\lfloor \frac{\deg g_1}{2} \rfloor}\|f\|.$$

‡‡‡. Les projetés ont une norme euclidienne plus petite.

◇◇◇. Voir par exemple [2].

En combinant avec l'inégalité précédente, on tombe sur la condition (8) et $\|b\|^n \|f\|^{n-1} < p^{kl}$: $\text{pgcd}(f, b_1)$ fournit un facteur non trivial.

Sinon, f est irréductible. ◀

On peut trouver une borne plus petite, mais l'objectif ici n'est pas l'efficacité, mais le fait de montrer que l'on est dans la classe **P** :

Proposition 5.3.2 *On peut trouver un facteur irréductible unitaire (différent de 1) de f en temps polynomial.*

Algorithme 3 FACTORISER

Entrée : $f \in \mathbb{Z}[X]$ *unitaire sans facteurs carrés.*
Sortie : *Un facteur irréductible non trivial de f.*

$r \leftarrow \text{RES}(f, f')$
$p \leftarrow q$ où q premier, $r \not\equiv 0[q]$
$g, h \leftarrow$BERLEKAMP(f, p) ▷ $f = gh[p]$
$u, v \leftarrow$EUCLIDE_ETENDU(g, h) ▷ $ug + vh = 1[p]$
$k \leftarrow \left\lceil \frac{1}{\deg g} \log_p \left(\|f\|^{2n-1} n^{\frac{n}{2}} \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}^n 2^{\frac{n(n-1)}{2}} \right) \right\rceil$
$l \leftarrow 0$
Tant que $2^l < k$ **Faire**
  $g, h, u, v \leftarrow$HENSEL(f, g, h, u, v)
  $l \leftarrow l + 1$
Fin Tant que
$R \leftarrow$réseau (4)
$B \leftarrow$LLL(R)
$g_0 \leftarrow \text{pgcd}(B_1, f)$
Si $g_0 = 1$ **Alors**
  Rendre f
Sinon
  Rendre g_0
Fin Si

Preuve :

▶ Montrons étape par étape que la complexité est polynomiale en les données.

Le calcul du résultant se fait par pivot de Gauss et sa valeur a un logarithme de l'ordre de $3n \log n + 2n(\log \|f\|)$.

Le calcul de p est lui aussi linéaire d'après la proposition 5.1.1. BERLEKAMP s'exécute en $O(p \times n^3)$, ce qui absorbe le temps demandé par EUCLIDE_ETENDU.

En utilisant la formule de Stirling, on voit que $k \times \log p = O(n^2 + n \log \|f\|)$: la boucle de remontée de HENSEL s'exécute $O(\log n + \log\log \|f\|)$ fois et son corps a une complexité en $O((k \log p)^2 n^2) = O(n^6 \log \|f\|^2)$ ce qui donne une complexité totale *** en $O(n^6 \log n \times \log \|f\|)$.

De toute façon, c'est LLL qui absorbe toute la complexité : en effet, son temps d'exécution est de l'ordre de $O(n^6 \times (k \log p)^3) = O(n^{12} + n^9 (\log \|f\|)^3)$. En effet, les vecteurs du réseau ont une norme de l'ordre de $\sqrt{n} p^k$ et ont donc leur logarithme en $O(k \log p)$ car $\log n$ est négligeable devant k. ◀

***. On assimile $\log\log \|f\|$ à une constante.

6 Comment accélérer l'algorithme

Ce qui précède montre un résultat théorique important, à savoir que le problème de la factorisation des polynômes de $\mathbb{Q}[X]$ est dans la classe **P**. Néanmoins, l'algorithme décrit est loin d'être efficace. En fait, les bornes utilisées dans la section 5 sont trop grandes pour espérer que les calculs, notamment ceux de l'orthogonalisation, soient réalisables rapidement. Voyons quelques améliorations qui accélèrent l'algorithme dans la plupart des cas.

6.1 Précision sur la complexité

Tout d'abord, notons que l'on peut baisser la complexité théorique annoncée en utilisant des algorithmes de multiplication rapide sur les entiers : multiplier deux entiers dont la taille en bits est de l'ordre de s peut être réduit à $O(s^{1+\varepsilon})$ pour tout $\varepsilon > 0$. Ceci permet de revoir la complexité de LLL à la baisse en $O(n^{5+\varepsilon}k^{2+\varepsilon})$ et donc une complexité totale en $O(n^{9+\varepsilon} + n^{7+\varepsilon}(\log \|f\|)^{2+\varepsilon})$. Néanmoins, cette amélioration n'est que théorique car l'efficacité de la multiplication rapide n'est atteinte que pour des entiers de l'ordre de 2^{1000}.

Notons que c'est la complexité au pire que nous avons calculée, en minorant le degré du facteur g par 1. Néanmoins, dans $\mathbb{F}_p[X]$ un polynôme de degré n a en moyenne $\log n$ facteurs irréductibles et on peut donc espérer un facteur g de degré $O\left(\frac{n}{\log n}\right) = O(n^{1-\varepsilon})$. Ceci permet d'obtenir une complexité en $O(n^{8+\varepsilon} + n^7(\log \|f\|)^{1+\varepsilon})$, qui est toujours entièrement déterminée par l'étape LLL.

Néanmoins, la complexité reste relativement élevée et en pratique, on peut trouver des astuces qui font chuter le temps de calcul dans la plupart des cas. La première est celle que l'on vient d'expliquer, qui consiste à chercher un facteur irréductible g pour des petits p tel que son degré soit le plus grand possible. Voyons quelques autres astuces.

6.2 Trouver un facteur non trivial

Deux questions se posent dans le problème de la factorisation : la première est de savoir si le polynôme est irréductible et la deuxième est de trouver un facteur non trivial pour pouvoir ensuite recommencer récursivement la factorisation. Il y a deux endroits dans l'algorithme présenté où l'on peut raisonnablement tester la divisibilité d'un facteur potentiel :

• Tout d'abord, lors de l'algorithme de relèvement de Hensel, il peut apparaître que le facteur considéré reste inchangé. Ainsi si pendant deux étapes du relèvement, le facteur n'est pas modifié, il est raisonnable qu'il s'agisse d'un facteur de f dans $\mathbb{Z}[X]$. Une stratégie de recherche d'un facteur irréductible consiste à relever chaque facteur un nombre de fois déterminé à l'avance [††††] et tester la divisibilité à chaque fois qu'un facteur est laissé inchangé pendant deux étapes.

• Lors du déroulement de LLL, il se peut que le premier vecteur devienne très petit par rapport aux autres : il y a des chances que ce petit vecteur ait en fait un pgcd avec f non trivial.

††††. 5 paraît être raisonnable.

6.3 Prouver l'irréductibilité

La deuxième question naturelle est de savoir si le polynôme de départ f est irréductible. Evidemment, si en le factorisant dans $\mathbb{F}_p$ on ne trouve qu'un facteur, alors il est clair qu'il est irréductible. Néanmoins, on n'a pas forcément cette chance. L'idée est de factoriser f dans plusieurs $\mathbb{F}_p$ et de confronter les différentes factorisations obtenues.

Prenons un exemple pour montrer l'utilité de la méthode : supposons qu'un polynôme de degré 20 se factorise d'une part en des polynômes irréductibles de degrés 6, 6 et 7 et d'autre part de degrés 4, 6 et 10. Cela nous indique que f a un facteur irréductible de degré au moins 10 mais la première factorisation impose qu'il soit de degré 12. En reprenant la deuxième factorisation on minore son degré par 14 puis avec la première par 20 : f est en fait irréductible.

6.4 Essayer des combinaisons

Avant l'utilisation de LLL, les algorithmes de factorisation commençaient de la même façon. Cependant, après la remontée de Hensel, on teste toutes les combinaisons des facteurs irréductibles modulo p^k afin de trouver ou non un « vrai » facteur du polynôme. D'un point de vue théorique, il peut y avoir 2^n combinaisons et l'algorithme est exponentiel. Néanmoins, on peut espérer qu'à cette étape, le polynôme soit « presque » irréductible et que le nombre de combinaisons soit beaucoup plus petit. Dans la pratique, il est souvent plus rapide d'essayer toutes les combinaisons plutôt que d'utiliser LLL.

6.5 Accélérer LLL

Néanmoins, il peut arriver que notre polynôme ait trop de facteurs irréductibles modulo les p que l'on a essayés. Il est alors sage d'appliquer LLL.

Le moyen le plus couramment utilisé pour accélérer LLL est d'une part de modifier l'orthogonalisée de la base au fur et à mesure de l'algorithme, sans tout recalculer à chaque étape et d'autre part d'effectuer les calculs d'orthogonalisation en mode flottant dans un premier temps puis de finir avec des calculs exacts.

Néanmoins, ceci est assez dangereux car les calculs en mode flottant peuvent conduire à faire boucler l'algorithme : imaginons que l'on ait à réaliser le produit scalaire entre un vecteur b_j^* dont les coordonnées soient toutes inférieures à 100 et un vecteur b_i dont les coordonnées soient de l'ordre de 10^{50}. Le calcul en mode flottant donne un résultat avec une précision de l'ordre de 10^{40} qui est très supérieur à la norme du petit vecteur. Ainsi pendant le calcul de $\mu_{ij} = \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)}$ on a seulement une précision que de l'ordre de 10^{40} et il est inconcevable de le comparer à $\frac{1}{2}$. Toutefois, on peut tourner cette situation à notre avantage car comme on l'a expliqué en 6.2, il se peut que $\text{pgcd}(f, b_j)$ soit non trivial.

Bibliographie

[1] A.K. Lenstra, H.W. Lenstra, L. Lovász. Factoring Polynomials with Rational Coefficients. In *Mathematische Annalen*, volume 261, pages 515–534. Springer 1982.

[2] C.K. Yap. *Fundamental Problems of Algorithmic Algebra.* Oxford University Press 2000.

[3] C.W.S. Cassels. *An introduction to the geometry of numbers.* Berlin, Heidelberg, New York : Springer 1971.

[4] J. Gathen, J. Gerhard. *Modern Computer Algebra.* Cambridge University Press 1999.