

ALGORITHME D'EUCLIDE MODULAIRE SUR LES POLYNÔMES

1. INTRODUCTION

Soient les polynômes

$$f(x) = 824x^5 - 65x^4 - 814x^3 - 741x^2 - 979x - 764$$

$$g(x) = 216x^4 + 663x^3 + 880x^2 - 916x + 617.$$

L'algorithme d'Euclide classique appliqué à f et g , fait apparaître des polynômes aux coefficients relativement grands. Ce fait n'est pas rare, et il y a lieu de se demander dans quelle mesure la taille des coefficients pénalise les performances de l'algorithme. En fait, une étude fine montre que l'algorithme s'exécute tout de même en temps polynomial en la taille des entrées.

Est-il possible de réduire, ou même éviter l'explosion des coefficients ? On peut tout d'abord exploiter le fait que le pgcd n'est défini qu'à une unité multiplicative près, et n'utiliser par exemple que des polynômes unitaires. C'est l'**algorithme d'Euclide unitaire**. On constate qu'avec cette variante, la taille des coefficients est plus petite.

On peut aussi s'arranger pour n'avoir que des polynômes dans $\mathbb{Z}[x]$, et donc faire à chaque étape une pseudo-division, c'est-à-dire multiplier le dividende par une puissance suffisante du coefficient dominant du diviseur. cela donne un algorithme (**algorithme d'Euclide primitif**) au moins aussi performant que l'algorithme unitaire.

Une autre possibilité est la *réduction modulo quelque chose*. Si comme dans l'exemple proposé on travaille dans $\mathbb{Z}[x]$, on peut réduire modulo un nombre premier bien choisi, calculer le pgcd des polynômes réduits dans $\mathbb{F}_p[x]$ et relever le résultat dans $\mathbb{Z}[x]$, si c'est possible.

Nous allons explorer ces diverses pistes.

2. VARIANTE «UNITAIRE» DE L'ALGORITHME D'EUCLIDE

Dans cette variante de l'algorithme d'Euclide, on divise à chaque étape le dernier reste par son coefficient dominant. Une analyse montre que cette variante n'est, au moins asymptotiquement, pas plus couteuse en nombre d'opérations sur le corps de base que l'algorithme classique.

Si on essaie le nouvel algorithme d'Euclide sur l'exemple de l'introduction, les coefficients qui apparaissent sont moins gros que quand on utilise l'algorithme classique. Et c'est ce qui se passe généralement, comme peut l'indiquer l'exécution des deux algorithmes sur plusieurs couples de polynômes choisis de façon aléatoire.

À titre de comparaison avec les algorithmes modulaires, indiquons le coût de cet algorithme unitaire, sur $\mathbb{Q}[x]$.

Théorème 2.1. *Soient f et g deux polynômes de $\mathbb{Z}[x]$, de degrés respectifs n et m , dont tous les coefficients sont bornés par A en valeur absolue, et soit δ l'écart maximum de degrés entre deux restes successifs dans l'algorithme d'Euclide unitaire appliqué à ces polynômes. Alors l'algorithme utilise $O(n^3 m \delta^2 \log^2(nA))$ opérations sur les mots.*

3. BORNE DE MIGNOTTE

Si l'on veut retrouver le pgcd de deux polynômes f et g de $\mathbb{Z}[x]$, connaissant le pgcd de leur réduction modulo un nombre premier p , il nous faut bien choisir le nombre premier p . Pour cela, nous allons établir une borne pour les coefficients de ce pgcd.

On définit la norme 2 d'un polynôme $f = \sum_{i=0}^n f_i x^i$ de $\mathbb{C}[x]$: $\|f\|_2 = (\sum_{i=0}^n |f_i|^2)^{1/2}$. On veut trouver une borne B pour la norme d'un facteur de f , c'est-à-dire telle que pour tout facteur h de f , on ait $\|h\|_2 \leq B$.

Lemme 3.1. *Pour $f \in \mathbb{C}[x]$ et $z \in \mathbb{C}$, on a $\|(x - z)f\|_2 = \|(\bar{z}x - 1)f\|_2$.*

Écrivons

$$f = \sum_{i=0}^n f_i x^i = f_n \prod_{i=1}^n (x - z_i).$$

On définit $M(f) = |f_n| \prod_{i=1}^n \max\{1, |z_i|\}$. Alors $M(f) \geq |\text{cd}(f)|$ et $M(f) = M(g)M(h)$ si $f = gh$. Le théorème suivant permet d'estimer $M(f)$:

Théorème 3.2 (Landau). *Pour tout $f \in \mathbb{C}[x]$, $M(f) \leq \|f\|_2$.*

Preuve. On peut ordonner les racines de f de façon que $|z_1|, \dots, |z_k| > 1$ et $|z_{k+1}|, \dots, |z_n| \leq 1$. Alors $M(f) = |f_n z_1 \dots z_k|$. Soit

$$g = f_n \prod_{i=1}^k (\bar{z}_i x - 1) \prod_{i=k+1}^n (x - z_i).$$

Alors

$$M(f)^2 = |f_n \bar{z}_1 \dots z_k| = |g_n|^2 \leq \|g\|_2^2.$$

Pour finir, par utilisations successives du lemme 3.1, on trouve que $\|g\|_2 = \|f\|_2$. \square

Pour la suite, on va également utiliser la norme 1 : $\|f\|_1 = \sum_{i=0}^n |f_i|$. Alors $\|f\|_\infty \leq \|f\|_2 \leq \|f\|_1 \leq (n+1)\|f\|_\infty$.

Théorème 3.3. *Soit $h = \sum_{i=0}^m h_i x^i \in \mathbb{C}[x]$ de degré m qui divise $f = \sum_{i=0}^n f_i x^i \in \mathbb{C}[x]$ de degré n . Alors*

$$\|h\|_2 \leq \|h\|_1 \leq 2^m M(h) \leq \left| \frac{h_m}{f_n} \right| 2^m M(f) \leq \left| \frac{h_m}{f_n} \right| 2^m \|f\|_2.$$

Preuve. La seconde inégalité vient de la règle de Viète décrivant les coefficients d'un polynôme en fonction de ses racines. La quatrième utilise le théorème 3.2. \square

On en déduit alors la **borne de Mignotte**.

Corollaire 3.4. *Soient f, g et h trois polynômes de $\mathbb{Z}[x]$ de degrés respectifs n, m et k , tels que gh divise f (dans $\mathbb{Z}[x]$). Alors*

- $\|g\|_\infty \|h\|_\infty \leq \|g\|_2 \|h\|_2 \leq \|g\|_1 \|h\|_1 \leq 2^{m+k} \|f\|_2 \leq (n+1)^{1/2} 2^{m+k} \|f\|_\infty$.
- $\|h\|_\infty \leq \|h\|_2 \leq 2^k \|f\|_2 \leq 2^{m+k} \|f\|_1$ et $\|h\|_\infty \leq \|h\|_2 \leq (n+1)^{1/2} 2^k \|f\|_\infty$.

4. ALGORITHME MODULAIRE : VERSION «GRAND NOMBRE PREMIER»

On considère deux polynômes primitifs f et g de $\mathbb{Z}[x]$. On va calculer leur pgcd h dans $\mathbb{Z}[x]$, en utilisant un algorithme *modulaire*, c'est-à-dire qu'on va réduire modulo un ou plusieurs nombres premiers.

Pour choisir ces nombres premiers, commençons par borner les coefficients du pgcd h . Soit A un majorant de $\|f\|_\infty$ et $\|g\|_\infty$. On suppose que le degré n de f est supérieur ou égal au degré de g . Alors $\|h\|_\infty \leq (n+1)^{1/2}2^n A$. On peut procéder de la façon suivante.

Soit b le pgcd de $\text{cd}(f)$ et $\text{cd}(g)$. Soit $B = (n+1)^{1/2}2^n Ab$. On choisit un nombre premier $p \in [2B, 4B]$ au hasard. Soient \bar{f} et \bar{g} les réductions de f et g modulo p . On utilise l'algorithme d'Euclide pour calculer le polynôme unitaire v de $\mathbb{Z}[x]$ tel que $\|v\|_\infty \leq p/2$ et tel que sa réduction modulo p est le pgcd de \bar{f} et \bar{g} . On calcule ensuite w , f^* et g^* dans $\mathbb{Z}[x]$ de norme infinie bornée par $p/2$ tels que

$$(1) \quad w \equiv bv \pmod{p}, \quad f^*w \equiv bf \pmod{p}, \quad g^*w \equiv bg \pmod{p}.$$

Alors, si

$$(2) \quad \|f^*\|_1 \|w\|_1 \leq B \quad \text{et} \quad \|g^*\|_1 \|w\|_1 \leq B$$

le pgcd de f et g est égal à $\text{pp}(w)$. Sinon, il faut recommencer avec un autre nombre premier.

Preuve. La condition (2) est en fait nécessaire et suffisante.

En effet, si cette condition est vérifiée, alors $\|f^*w\|_\infty \leq p/2$. Comme $\|bf\|_\infty \leq p/2$ et $f^*w \equiv bf \pmod{p}$, on en déduit que $f^*w = bf$. De même, $g^*w = bg$. Donc $w \mid \text{pgcd}(bf, bg)$. Alors, un raisonnement sur les degrés, et le fait que $\bar{h} \mid \text{pgcd}(\bar{f}, \bar{g})$ montrent que $\text{pp}(w) = \text{pgcd}(f, g)$.

Réciproquement, si $\text{pp}(w) = \text{pgcd}(f, g)$, alors w divise bf . D'après la borne de Mignotte, $\|bf/w\|_\infty \leq B \leq p/2$. Donc $f^* = bf/w$ et de même, $g^* = bg/w$. Donc, le corollaire (3.4) montre que la condition (2) est vérifiée. \square

Théorème 4.1. *L'algorithme précédent utilise $O(n^2(n^2 + \log^2 A))$ opérations sur les mots.*

Il reste le problème de trouver le nombre premier par lequel on va réduire.

Théorème 4.2. *Soit $r = \text{Res}(f/h, g/h)$. L'entier r est non nul, majoré en valeur absolue par $(n+1)^n A^{2n}$. De plus, la condition (2) est vraie si et seulement si $p \nmid r$. Enfin, il existe un algorithme probabiliste utilisant $O(n^3 + \log^3 A)$ opérations sur les mots, déterminant un entier p entre $2B$ et $4B$ tel que p est premier ne divisant pas r avec probabilité au moins $1/2$. La moyenne du nombre d'itérations nécessaires est donc au plus 2.*

5. ALGORITHME MODULAIRE : VERSION «PETITS NOMBRES PREMIERS»

On peut modifier cet algorithme en réduisant modulo plusieurs petits nombres premiers, et en relevant les résultats par l'utilisation du théorème chinois. On doit alors à chaque pas effectuer plusieurs fois l'algorithme d'Euclide, mais les coefficients qui interviennent sont plus petits, ce qui diminue considérablement le temps d'exécution pour de grandes valeurs des degrés et des normes infinies des polynômes f et g de départ.

Soient deux polynômes primitifs f et g de $\mathbb{Z}[x]$, tels que $\deg(f) = n \geq \deg(g)$ et de norme infinie inférieure ou égale à A .

On pose $b = \text{pgcd}(\text{cd}(f), \text{cd}(g))$, $k = \lceil 2 \log_2((n+1)^n b A^{2n}) \rceil$, $B = (n+1)^{1/2} 2^n A b$ et $L = \lceil \log_2(2B+1) \rceil$.

On choisit un ensemble S de $2L$ nombres premiers inférieurs à $2k \log k$, et ne divisant pas b . Pour chaque p dans S , on calcule le polynôme unitaire v_p à coefficients dans $\{0, \dots, p-1\}$ dont la réduction modulo p est égale au pgcd des réductions \bar{f} et \bar{g} modulo p . Soit $e = \min \{\deg(v_p) : p \in S\}$, on remplace $S \leftarrow \{p \in S : \deg(v_p) = e\}$. Si le cardinal $|S| \geq L$, alors on enlève $|S| - L$ éléments de S , sinon, on recommence avec un nouveau choix de S .

On calcule, grâce au théorème chinois, les polynômes w , f^* et g^* de $\mathbb{Z}[x]$ de norme infinie inférieure à $(\prod_{p \in S} p)/2$ tels que pour tout $p \in S$

$$(3) \quad w \equiv b v_p \pmod{p}, \quad f^* w \equiv b f \pmod{p}, \quad g^* w \equiv b g \pmod{p}.$$

Alors si

$$(4) \quad \|f^*\|_1 \|w\|_1 \leq B \quad \text{et} \quad \|g^*\|_1 \|w\|_1 \leq B,$$

on a $\text{pp}(w) = \text{pgcd}(f, g)$. Sinon, il faut recommencer.

Théorème 5.1. *L'algorithme utilise $O(n(n^2 + \log^2 A)(\log n + \log \log A)^2)$ opérations sur les mots.*

Référence :

Modern computer algebra, Joachim von zur Gathen, Jürgen Gerhard. Chapitre 6.