@languagefrench
|

Algorithmes de calcul de PGCD

B. Parisse
Institut Fourier
UMR 5582 du CNRS
Université de Grenoble I

14th July 2002

Comme on l'a remarqué dans le premier article, l'algorithme d'Euclide est inefficace pour calculer le pgcd de deux polynômes à coefficients entiers. On va présenter ici les algorithmes utilisés habituellement par les systèmes de calcul formel: sous-résultant (PRS), modulaire (GCDMOD), p-adique (EEZGD) et heuristique (GCDHEU). Le premier est une adaptation de l'algorithme d'Euclide et s'adapte à des coefficients assez génériques. Les trois autres ont en commun d'évaluer une ou plusieurs variables du polynôme (dans ce dernier cas il est nécessaire de bien distinguer le cas de polynômes à plusieurs variables) et de reconstruire le pgcd par des techniques distinctes, la plupart du temps ces algorithmes fonctionnent seulement si les coefficients sont entiers.

Soit donc P et Q deux polynômes à coefficients dans un corps. Le pgcd de P et Q n'est défini qu'à une constante près. Mais lorsque les coefficients de P et Q sont dans un anneau euclidien comme par exemple $\mathbb{Z}$ ou $\mathbb{Z}[i]$, on appelle pgcd de P et Q un polynôme D tel que P/D et Q/D soient encore à coefficients dans l'anneau, et que D soit optimal, c'est-à-dire que si un multiple μD de D vérifie $P/\mu D$ et $Q/\mu D$ sont à coefficients dans l'anneau, alors μ est inversible. La première étape d'un algorithme de calcul de pgcd consiste donc à diviser par le pgcd des coefficients entiers de chaque polynôme.

Exemple: $P = 4X^2 - 4$ et $Q = 6X^2 + 12X + 6$. Le polynôme $X + 1$ est un pgcd de P et Q puisqu'il est de degré maximal divisant P et Q mais le pgcd de P et Q est $2(X + 1)$. Remarquons qu'avec notre définition $-2(X + 1)$ convient aussi. Par convention on appelera pgcd le polynôme ayant un coefficient dominant positif.

Définition: On appelle contenu $c(P)$ d'un polynôme P le pgcd des coefficients de P. On définit alors la partie primitive de P: $\mathrm{pp}(P) = P/c(P)$. Si $c(P) = 1$, on dit que P est primitif. On montre que :

$$D = \mathrm{pgcd}(P, Q) = \mathrm{pgcd}(c(P), c(Q))\,\mathrm{pgcd}(\mathrm{pp}(P), \mathrm{pp}(Q))$$

1 Le sous-résultant.

La première idée qui vient à l'esprit pour améliorer l'efficacité de l'algorithme d'Euclide consiste à éviter les fractions qui sont créées par les divisions euclidiennes. On utilise à cet effet la pseudo-division: au lieu de prendre le reste R de la division euclidienne du polynôme P par Q, on prend le reste de la division de $Pq^{\delta+1}$ par Q, où q désigne le coefficient dominant de Q et δ la différence entre le degré de P et de Q.

Exercice: En utilisant votre système de calcul formel préféré, calculez les restes intermédiaires générés dans l'algorithme d'Euclide lorsqu'on utilise la pseudo-division par exemple pour les polynômes $P(x) = (x+1)^7 - (x-1)^6$ et sa dérivée.

Une solution avec giac/xcas:

```
// -*- mode:C++ -*- a,b 2 polynomes -> pgcd de a et b
pgcd(a,b):={
 local P,p,Q,q,R,g,h,d;
 // convertit a et b en polynomes listes et extrait la partie primitive
 P:=symb2poly1(a);
 p:=lgcd(P); // pgcd des elements de la liste
 P:=P/p;
 Q:=symb2poly1(b);
 q:=lgcd(Q);
 Q:=Q/q;
 if (size(P)<size(Q)){ // echange P et Q
  R:=P; P:=Q; Q:=R;
 }
 // calcul du contenu du pgcd
 p:=gcd(p,q);
 g:=1;
 h:=1;
 while (size(Q)!=1){
  q:=Q[0]; // coefficient dominant
  d:=size(P)-size(Q);
  R:=rem(q^(d+1)*P,Q);
  if (size(R)==0) return(p*poly12symb(Q/lgcd(Q),x));
  P:=Q;
  Q:=R;
  // ligne suivante a decommenter pour prs
  // Q:=R/(g*h^d);
  print(Q);
  // ligne suivante a decommenter pour prs
  // g:=q; h:=q^d/h^(d-1);
 }
 return(p);
}
```

On s'aperçoit que les coefficients croissent de manière exponentielle. La deuxième idée qui vient naturellement est alors à chaque étape de rendre le reste primitif, donc de diviser R par le pgcd de ces coefficients. Cela donne un algorithme plus efficace, mais encore assez peu efficace car à chaque étape on doit calculer le pgcd de tous les coefficients, on peut imaginer le temps que cela prendra en dimension 1 et à fortiori en dimension supérieure. L'idéal serait de connaitre à l'avance une quantité suffisamment grande qui divise tous les coefficients du reste.

C'est ici qu'intervient l'algorithme du sous-résultant: après chaque pseudo-division euclidienne, on exhibe un coefficient "magique" qui divise les coefficients du reste. Ce coefficient n'est pas le pgcd mais il est suffisamment grand pour qu'on évite la croissance exponentielle des coefficients.

Algorithme du sous-résultant

Arguments: 2 polynômes P et Q primitifs. Valeur de retour: le pgcd de P et Q.

Pour calculer le coefficient "magique" on utilise 2 variables auxiliaires g et h initialisées a 1.

Boucle à effectuer tant que Q est non nul:

- on note δ =degre(P)-degre(Q) et q le coefficient dominant de Q
- on effectue la division euclidienne (sans fraction) de $q^{\delta+1}P$ par Q, soit R le reste
- Si R est constant, on sort de l'algorithme en renvoyant 1 comme pgcd
- on recopie Q dans P puis $R/(gh^\delta)$ dans Q
- on recopie q dans g et $h^{1-\delta}q^\delta$ dans h.

Si on sort normalement de la boucle, Q est nul, on renvoie donc la partie primitive de P qui est le pgcd cherché.

Pour tester l'algorithme avec `xcas`, il suffit de décommenter les deux lignes `Q:=R/(g*h^d);` et `g:=q; h:=q^d/h^(d-1);` ci-dessus.

La preuve de l'algorithme est un peu longue et par ailleurs très bien faite dans le 2ème tome de Knuth (The Art of Computer Programming, Semi-numerical Algorithms), on y renvoie donc le lecteur intéressé. L'idée générale (et le nom de la méthode) est de considérer la matrice de Sylvester des polynômes de départ P et Q (celle dont le déterminant est appelé résultant de P et Q) et de traduire les pseudo-divisions qui permettent de calculer les restes successifs du sous-résultant en opération de ligne sur ces matrices. On démontre alors que les coefficients de R divisés par gh^δ peuvent être interprétés comme des déterminants de sous-matrices de la matrice de Sylvester après réduction et c'est cela qui permet de conclure qu'ils sont entiers.

2 Le pgcd en une variable

.

2.1 Le pgcd heuristique.

On suppose ici que les coefficients sont entiers ou entiers de Gauss. **On peut donc se ramener au cas où les polynômes sont primitifs.**

L'idée consiste à évaluer P et Q en un entier z et à extraire des informations du pgcd g des entiers $P(z)$ et $Q(z)$. Il faut donc un moyen de remonter de l'entier g à un polynôme G tel que $G(z) = g$. La méthode consiste à écrire en base z l'entier g, avec une particularité dans les divisions euclidiennes successives on utilise le reste symétrique (compris entre $-z/2$ et $z/2$). Cette écriture donne les coefficients d'un polynôme G unique. On extrait ensuite la partie primitive de ce polynôme G. Lorsque z est assez grand par rapport aux coefficients des polynômes P et Q, si $\mathrm{pp}(G)$ divise P et Q, on va montrer que le pgcd de P et de Q est $D = \mathrm{pp}(G)$.

On remarque tout d'abord que $d := D(z)$ divise g. En effet D divise P et Q donc pour tout entier (ou entier de Gauss) z, $D(z)$ divise $P(z)$ et $Q(z)$. Il existe donc une constante a telle que

$$g = ad$$

On a aussi $\mathrm{pp}(G)$ divise D. Il existe donc un polynôme C tel que :

$$D = \mathrm{pp}(G)C$$

Nous devons prouver que C est un polynôme constant. On suppose dans la suite que ce n'est pas le cas. Evaluons l'égalité précédente au point z, on obtient

$$d = \frac{g}{c(G)} C(z)$$

Finalement

$$1 = \frac{a}{c(G)} C(z)$$

La procédure de construction de G nous donne une majoration de ces coefficients par $|z|/2$, donc de $c(G)$ par $|z|/2$, donc $C(z)$ divise un entier de module plus petit que $|z|/2$, donc

$$|C(z)| \leqslant \frac{|z|}{2}$$

On considère maintenant les racines complexes $z_1, \ldots, z_n$ du polynôme C (il en existe au moins une puisqu'on a supposé C non constant). On a:

$$C(X) = c_n(X - z_1) \ldots (X - z_n)$$

Donc, comme c_n est un entier (ou entier de Gauss) non nul, sa norme est supérieure ou égale à 1 et :

$$|C(z)| \geqslant \prod_{j=1}^{n} (|z| - |z_j|)$$

Il nous reste à majorer les racines de C pour minorer $|C(z)|$. Comme C divise D il divise P et Q donc les racines de C sont des racines communes à P et Q. On va appliquer le:

Lemme 1 *Soit x une racine complexe d'un polynôme $P = a_n X^n + \dots. + a_0$. Alors*

$$|x| < \frac{|P|}{|a_n|} + 1, |P| = \max_{0 \leqslant i \leqslant n} (|a_i|)$$

Application du lemme à $C(X)$: on a $1/|c_n| \leq 1$ donc si on a choisi z tel que $|z| \geqslant 2\min(|P|, |Q|) + 2$, alors pour tout j, $|z_j| < |z|/2$ donc

$$|C(z)| > \left(\frac{|z|}{2}\right)^n$$

qui contredit notre majoration de $|C(z)|$.

Théorème 1 *Soit P et Q deux polynômes à coefficients entiers. On choisit un entier z tel que $|z| \geqslant 2\min(|P|, |Q|) + 2$, si la partie primitive du polynôme G reconstruit à partir du pgcd de $P(z)$ et $Q(z)$ par écriture en base z (avec comme reste euclidien le reste symétrique) divise P et Q alors c'est le pgcd de P et Q.*

Pour finir la démonstration du théorème, il nous faut encore montrer le lemme. On a

$$-a_n x^n = a_{n-1}x^{n-1} + \dots. + a_0$$

Donc

$$|a_n||x|^n \leqslant |P|(1 + \dots. + |x|^{n-1}) = |P|\frac{|x|^n - 1}{|x| - 1}$$

Ici on peut supposer que $|x| \geqslant 1$, sinon le lemme est démontré, donc $|x| - 1$ est positif et

$$|a_n|(|x| - 1) \leqslant |P|\frac{|x|^n - 1}{|x|^n} \Rightarrow |x| - 1 < \frac{|P|}{|a_n|}$$

Remarques

- Le théorème publié par Char, Geddes et Gonnet porte sur des coefficients entiers et c'est comme cela qu'il est utilisé par les systèmes de calcul formel (en commençant historiquement par Maple). Peu de systèmes l'utilisent pour les polynômes à coefficients entiers de Gauss. On peut d'ailleurs généraliser le théorème à d'autres types de coefficients, à condition d'avoir un anneau euclidien plongé dans $\mathbb{C}$ avec une minoration sur la valeur absolue des élements non nuls de l'anneau.

- Nous n'avons jusqu'à présent aucune certitude qu'il existe des entiers z tels que la partie primitive de G divise P et Q. Nous allons montrer en utilisant l'identité de Bézout que pour z assez grand c'est toujours le cas. Plus précisément, on sait qu'il existe deux polynômes U et V tels que

  $$PU + QV = D$$

  Attention toutefois, U et V sont à coefficients rationnels, pour avoir des coefficients entiers, on doit multiplier par une constante entière α, donc en évaluant en z on obtient l'existence d'une égalité à coefficients entiers

  $$P(z)u + Q(z)v = \alpha D(z)$$

Donc le pgcd g de $P(z)$ et $Q(z)$ divise $\alpha D(z) = \alpha d$. Comme g est un multiple de d, on en déduit que $g = \beta d$, où β est un diviseur de α. Si on a choisi z tel que

$$|z| > \ 2|D||\alpha|$$

alors $|z| > 2|D||\beta|$ donc l'écriture symétrique en base z de g est $G = \beta D$. Donc la partie primitive de G est D, le pgcd de P et Q.

Exemple 1 Si $P_0 = 6(X^2 - 1)$ et $Q_0 = 4(X^3 - 1)$.

Le contenu de P_0 est 6, celui de Q_0 est 4.

On a donc pgcd des contenus = 2, $P = X^2 - 1, Q = X^3 - 1$. La valeur initiale de z est donc $2 * 1 + 2 = 4$. On trouve $P(4) = 15, Q(4) = 63$. Le pgcd entier de 15 et 63 est 3 que nous écrivons symétriquement en base 4 sous la forme $3 = 1 * 4 - 1$, donc $G = X - 1$, sa partie primitive est $X - 1$. On teste si $X - 1$ divise P et Q, c'est le cas, donc c'est le pgcd de P et Q et le pgcd de P_0 et Q_0 est $2(X - 1)$.

Algorithme gcdheu

En arguments deux polynômes P_0 et Q_0 à coefficients entiers ou entiers de Gauss. Retourne le pgcd de P_0 et Q_0 ou faux en cas d'échec.

1. Calculer le contenu de P_0 et Q_0. Vérifier que les coefficients sont entiers de Gauss sinon retourner faux.
2. Extraire la partie primitive P de P_0 et Q de Q_0, calculer le pgcd c des contenus de P_0 et Q_0
3. Déterminer $z = 2\min(|P|, |Q|) + 2$.
4. Début de boucle: initialisation du nombre d'essais à 1, test d'arrêt sur un nombre maximal d'essais, avec changement de z entre deux itérations (par exemple $z := 2z$).
5. Calculer le pgcd g de $P(z)$ et $Q(z)$ puis son écriture symétrique en base z dont on extrait la partie primitive G.
6. Si G ne divise pas$P$ passer à l'itération suivante. De même pour Q.
7. Retourner cG
8. Fin de la boucle
9. Retourner faux.

On remarque au passage qu'on a calculé le quotient de P par G et le quotient de Q par G lorsque la procédure réussit. On peut donc passer à la procédure gcdheu deux paramètres supplémentaires par référence, les deux polynômes que l'on affectera en cas de succès, ce qui optimise la simplification d'une fraction de 2 polynômes.

2.2 Le pgcd modulaire

On part du fait que si D est le pgcd de P et Q dans $\mathbb{Z}$ (ou $\mathbb{Z}[i]$) alors après réduction modulo un nombre premier n qui ne divise pas les coefficients dominants de P et Q, D divise le pgcd G de P et Q dans $\mathbb{Z}/n\mathbb{Z}$ (par convention, le pgcd dans $\mathbb{Z}/n\mathbb{Z}$ est normalisé pour que son coefficient dominant vaille 1). Comme on calcule G dans $\mathbb{Z}/n\mathbb{Z}$, les coefficients des restes intermédiaires de l'algorithme d'Euclide sont bornés, on évite ainsi la croissance exponentielle des coefficients. Il faudra ensuite reconstruire D à partir de G.

On remarque d'abord que si on trouve $G = 1$, alors P et Q sont premiers entre eux. En général, on peut seulement dire que le degré de G est supérieur ou égal au degré de D. En fait, le degré de G est égal au degré de D lorsque les restes de l'algorithme d'Euclide (calculé en effectuant des pseudo-divisions, cf. l'exercice 1) ont leur coefficient dominant non divisible par n. Donc plus n est grand, plus la probabilité est grande de trouver G du bon degré.

Dans la suite, nous allons déterminer une borne b à priori majorant les coefficients de D. On utilisera ensuite la même méthode que dans l'algorithme modulaire de recherche de racines évidentes: on multiplie G dans $\mathbb{Z}/n\mathbb{Z}$ par le pgcd dans $\mathbb{Z}$ des coefficients dominants p et q de P et Q. Soit $\tilde{D} = \text{pgcd}(p, q)G$ le résultat écrit en représentation symétrique. Si $n \geqslant b\,\text{pgcd}(p, q)$ et si G est du bon degré, on montre de la même manière que $D = \tilde{D}$. Comme on ne connait pas le degré de D, on est obligé de tester si $\tilde{D}$ divise P et Q. Si c'est le cas, alors $\tilde{D}$ divise D donc $\tilde{D} = D$ puisque $\text{degre}(\tilde{D}) = \text{degre}(G) \geqslant \text{degre}(D)$. Sinon, n est un nombre premier malchanceux pour ce calcul de pgcd ($\text{degre}(G) \geqslant \text{degre}(D)$), il faut essayer un autre premier.

Remarque: On serait tenté de dire que les coefficients de D sont bornés par le plus grand coefficient de P. C'est malheureusement faux, par exemple $(X + 1)^2$ dont le plus grand coefficient est 2 divise $(X + 1)^2(X - 1)$ dont le plus grand coefficient (en valeur absolue) est 1.

Soit $P = \sum p_i X^i$ un polynôme à coefficients entiers. On utilise la norme euclidienne

$$|P|^2 = \sum |p_i|^2 \tag{1}$$

On établit d'abord une majoration du produit des racines de norme supérieure à 1 de P à l'aide de $|P|$. Ensuite si D est un diviseur de P, le coefficient dominant d de D divise le coefficient dominant p de P et les racines de D sont aussi des racines de P. On pourra donc déterminer une majoration des polynômes symétriques des racines de D et donc des coefficients de D.

Lemme 2 *Soit $A = \sum_{j=0}^{a} a_j X^j$ un polynôme et $\alpha \in \mathbb{C}$. Alors*

$$|(X - \alpha)A| = |(\overline{\alpha}X - 1)A|$$

Pour prouver le lemme 2, on développe les produits de polynômes. On pose $a_{-1} = a_{a+1} = 0$ et on note $\Re$ la partie réelle.

$$|(X-\alpha)A|^2 = \sum_{j=0}^{a+1} |a_{j-1} - \alpha a_j|^2 = \sum_{j=0}^{a+1} |a_{j-1}|^2 + |\alpha|^2|a_j|^2 - 2\Re(a_{j-1}\overline{\alpha a_j})$$

$$|(\overline{\alpha}X-1)A|^2=\sum_{j=0}^{a+1}|\overline{\alpha}a_{j-1}-a_j|^2=\sum_{j=0}^{a+1}|\alpha|^2|a_{j-1}|^2+|a_j|^2-2\Re(\overline{\alpha}a_{j-1}\overline{a_j})$$

Les deux donnent bien le même résultat.

Soit $P(X)=p\prod(X-\alpha_j)$ la factorisation de P sur $\mathbb{C}$. On introduit le polynôme

$$\tilde{P}=p\prod_{j/|\alpha_j|\geqslant 1}(X-\alpha_j)\prod_{j/|\alpha_j|<1}(\overline{\alpha_j}X-1)$$

qui d'après le lemme a la même norme que P. La norme de P majore donc le coefficient constant de $\tilde{P}$ d'où:

$$\prod_{j/|\alpha_j|\geqslant 1}|\alpha_j|\leqslant\frac{|P|}{|p|}\tag{2}$$

On remarque que (2) reste vraie si on considère les racines δ_j de norme plus grande que 1 d'un diviseur D de P puisque le produit porte alors sur un sous-ensemble. On écrit maintenant l'expression des coefficients d_j de D à l'aide des racines δ_j de D:

$$|d_{m-j}|=|d|\left|\sum_{\text{choix de } j \text{ racines parmi les } m \text{ racines de } D}\ \prod_{\delta_k\in\text{racines choisies}}\delta_k\right|$$

Pour majorer $|d_{m-j}|$, on commence par majorer $|\delta_k|$ par $\beta_k=\max(1,|\delta_k|)$. On est donc ramené à majorer

$$\sigma_{j,m}(\beta)=\sum_{\text{choix de } j \text{ parmi } m \text{ valeurs } \beta_k}\ \prod_{\beta_k\in\text{choix}}\beta_k$$

avec pour hypothèse une majoration de $M=\prod_{k=1}^m\beta_k$ donnée par la relation (2). Pour cela, on cherche le maximum de $\sigma_{j,m}(\beta)$ sous les contraintes M fixé et $\beta_k\geqslant 1$.

On va montrer que le maximum ne peut être atteint que si l'un des $\beta_k=M$ (et tous les autres $\beta_k=1$). Sinon, quitte à réordonner supposons que les β_k sont classés par ordre croissant. On a donc $\beta_{m-1}\neq 1$, on pose $\widetilde{\beta_k}=\beta_k$ pour $k\leqslant m-2$, $\tilde{\beta}_{m-1}=1$ et $\tilde{\beta}_m=\beta_{m-1}\beta_m$. Comparons $\sigma_{j,m}(\beta)$ et $\sigma_{j,\mathrm{nm}}(\tilde{\beta})$. Si le choix de j parmi m comporte $k=m-1$ et $k=m$, le produit est inchangé. Sinon on a la somme de deux produits, l'un contenant $k=m-1$ et l'autre $k=m$. On compare donc $B(\beta_{m-1}+\beta_m)$ et $B(1+\beta_{m-1}\beta_m)$ avec $B=\prod_{\beta_k\in\text{reste du choix}}\beta_k$. Comme

$$1+\beta_{m-1}\beta_m\geqslant\beta_{m-1}+\beta_m$$

puisque la différence est le produit $(1-\beta_m)(1-\beta_{m-1})$ de deux nombres positifs, on arrive à la contradiction souhaitée.

Ensuite on décompose les choix de $\sigma_{m,j}$ en ceux contenant M et des 1 et ceux ne contenant que des 1, d'où la majoration

$$\sigma_{j,m}(\beta)\leqslant\left(\begin{array}{c}m-1\\ j-1\end{array}\right)M+\left(\begin{array}{c}m-1\\ j\end{array}\right)$$

et finalement

$$|d_{m-j}| \leqslant |d| \left(\left(\begin{array}{c} m-1 \\ j-1 \end{array} \right) \frac{|P|}{|p|} + \left(\begin{array}{c} m-1 \\ j \end{array} \right) \right) \qquad (3)$$

On peut en déduire une majoration indépendante de j sur les coefficients de D, en majorant $|d|$ par $|p|$ (puisque d divise p) et les coefficients binomiaux par 2^{m-1} (obtenue en développant $(1+1)^{m-1}$). D'où le

Théorème 2 *(Landau-Mignotte) Soit P un polynôme à coefficients entiers (ou entiers de Gauss) et D un diviseur de P de degré m. Si $|P|$ désigne la norme euclidienne du vecteur des coefficients de P et p le coefficient dominant de P alors les coefficients d_j de D satisfont l'inégalité*

$$|d_j| \leqslant 2^{m-1}(|P| + |p|) \qquad (4)$$

Avec cette estimation, on en déduit que si n est un premier plus grand que

$$\min \left(2^{\mathrm{degre}(P)-1}(|P| + |p|), 2^{\mathrm{degre}(Q)-1}(|Q| + |q|) \right), \qquad (5)$$

alors le pgcd trouvé dans $\mathbb{Z}/n\mathbb{Z}$ va se reconstruire en un pgcd dans $\mathbb{Z}$ si son degré est le bon.

Malheureusement la borne précédente est souvent très grande par rapport aux coefficients du pgcd et calculer dans $\mathbb{Z}/n\mathbb{Z}$ s'avèrera encore inefficace (surtout si le pgcd est 1). Cela reste vrai même si on optimise un peu la majoration (5) en repartant de (3).

L'idée est donc de travailler modulo plusieurs nombres premiers plus petits et reconstruire le pgcd des 2 polynômes à coefficients entiers à partir des pgcd des polynômes dans $\mathbb{Z}/n\mathbb{Z}$ et du théorème des restes chinois. En pratique on prend des nombres premiers inférieurs à la racine carrée du plus grand entier hardware de la machine (donc plus petits que 2^{16} sur une machine 32 bits) ce qui permet d'utiliser l'arithmétique hardware du processeur sans risque de débordement.

Algorithme du PGCD modulaire en 1 variable:

En argument: 2 polynômes primitifs P et Q à coefficients entiers. Le résultat renvoyé sera le polynôme pgcd.

Variable auxiliaire: un entier N initialisé à 1 qui représente le produit des nombres premiers utilisés jusqu'ici et un polynôme H initialisé à 0 qui représente le pgcd dans $\mathbb{Z}/N\mathbb{Z}$.

Boucle infinie :

1. Chercher un nouveau nombre premier n qui ne divise pas les coefficients dominants p et q de P et Q

2. Calculer le pgcd G de P et Q dans $\mathbb{Z}/n\mathbb{Z}$. Si G=1, renvoyer 1.

3. Si $H = 0$ ou si le degré de G est plus petit que le degré de H, recopier G dans H et n dans N, passer à la 6ème étape

4. Si le degré de G est plus grand que celui de H passer à l'itération suivante
5. Si le degré de G est égal au degré de H, en utilisant le théorème des restes chinois, calculer un polynôme $\tilde{H}$ tel que $\tilde{H} = H$ modulo N et $\tilde{H} = G$ modulo n. Recopier $\tilde{H}$ dans H et nN dans N.
6. Ecrire $\text{pgcd}(p, q)H$ en représentation symétrique. Soit $\tilde{H}$ le résultat rendu primitif. Tester si $\tilde{H}$ divise P et Q. Si c'est le cas, renvoyer $\tilde{H}$, sinon passer à l'itération suivante.

Finalement on n'a pas utilisé b, la borne de Landau-Mignotte. On peut penser que l'étape 6 ne devrait être effectuée que lorsque N est plus grand que $\text{pgcd}(p, q)b$. En pratique, on effectue le test de l'étape 6 plus tôt parce que les coefficients du pgcd sont rarement aussi grand que b. Mais pour éviter de faire le test trop tôt, on introduit une variable auxiliaire H' qui contient la valeur de H de l'itération précédente et on ne fait le test que si $H' = H$ (ou bien sûr si on a dépassé la borne).

Remarque:

L'algorithme ci-dessus fonctionne également pour des polynômes à plusieurs variables.

Exemple 1:

Calcul du pgcd de $(X+1)^3(X-1)^4$ et (X^4-1). Prenons pour commencer $n = 2$. On trouve comme pgcd X^4+1 (en effet $-1 = 1$ donc on cherchait le pgcd de $(X+1)^7$ et de $X^4+1 = (X+1)^4$). On teste si X^4+1 divise P et Q, ce n'est pas le cas donc on passe au nombre premier suivant. Pour $n = 3$, on trouve X^2-1. Donc $n = 2$ n'était pas un bon nombre premier pour ce calcul de pgcd puisqu'on a trouvé un pgcd de degré plus petit. On teste si X^2-1 divise P et Q, c'est le cas ici donc on peut arrêter, le pgcd cherché est X^2-1.

Exemple 2 :

Calcul du pgcd de $(X+1)^3(X-1)^4$ et $(X^4-1)^3$. Pour $n = 2$, on trouve un polynôme de degré 7. Pour $n = 3$, on trouve X^6-1 donc $n = 2$ était une mauvaise réduction. Comme X^6-1 ne divise pas P et Q, on passe à $n = 5$. On trouve $X^6+2X^4-2X^2-1$. On applique le théorème des restes chinois qui va nous donner un polynôme dans $\mathbb{Z}/15\mathbb{Z}$. On cherche donc un entier congru à 2 modulo 5 et à 0 modulo 3, -3 est la solution (écrite en représentation symétrique), donc le polynôme modulo 15 est $X^6-3X^4+3X^2-1 = (X^2-1)^3$. Ce polynôme divise P et Q, c'est donc le pgcd de P et de Q.

3 Le pgcd à plusieurs variables.

3.1 Le pgcd heuristique.

On suppose comme dans le cas à une variable que les polynômes sont primitifs, donc qu'on a simplifié les polynômes par le pgcd entier de leurs coefficients entiers.

Le principe est identique à celui du PGCD à 1 variable, on évalue les deux polynômes P et Q de k variables $X_1, \ldots, X_k$ en un $X_k = z$ et on calcule le pgcd

g des 2 polynômes $P(z)$ et $Q(z)$ de $k-1$ variables. On remonte ensuite à un polynôme G par écriture symétrique en base z de g et on teste si $\mathrm{pp}(G)$ divise P et Q. Il s'agit à nouveau de montrer que si z est assez grand, alors $\mathrm{pp}(G)$ est le pgcd cherché. On sait que $d = D(z)$ divise g. Il existe donc un polynôme a de $k-1$ variables tel que $g = ad$. On sait aussi que $\mathrm{pp}(G)$ divise D, donc il existe un polynôme C de k variables tel que $D = C * \mathrm{pp}(G)$. On évalue en z et on obtient $d = C(z)g/c(G)$, où $c(G)$ est un entier, donc

$$c(G) = a * C(z)$$

Comme $c(G)$ est un entier, a et $C(z)$ sont des polynômes constants. Comme précédemment, on a aussi $|C(z)| \leqslant |z|/2$ puisque $|c(G)| \leqslant |z|/2$.

- Premier cas: si C ne dépend que de la variable X_k. On continue le raisonnement comme dans le cas unidimensionnel.

- Deuxième cas: si C dépend d'une autre variable, par exemple X_1. On regarde le coefficient de plus haut degre de C par rapport a X_1. Ce coefficient divise le coefficient de plus haut degre de P et de Q par rapport a X_1. Comme $C(z)$ est constant, on en deduit que le coefficient de plus haut degre de P et Q par rapport a X_1 est divisible par $X_k - z$ donc le coefficient de plus bas degre en X_k de ces coefficients de plus haut degre est divisible par z, ce qui contredit la majoration de ce coefficient.

En pratique, cet algorithme nécessite le calcul récursif de pgcd sans garantie de réussite. On l'évite donc s'il y a beaucoup de variables (la limite est par exemple de 5 pour MuPAD).

3.2 Le pgcd modulaire multivariables.

Ici, on travaille modulo $X_n - \alpha$, où $X_1, \ldots, X_n$ désignent les variables des polynômes. On considère donc deux polynômes P et Q comme polynômes de la variables X_n avec des coefficients dans $\mathbb{Z}[X_1, \ldots, X_{n-1}]$. On évalue en $X_n = \alpha$, on obtient deux polynômes en $n-1$ variables dont on calcule le pgcd (récursivement).

Il s'agit de reconstruire le pgcd par interpolation. Tout d'abord, on a une borne évidente sur le degré du pgcd par rapport à la variable X_n, c'est le minimum δ des degrés par rapport à X_n des polynômes P et Q. A première vue, il suffit donc d'évaluer les polynômes en $\delta + 1$ points α.

Il faut toutefois prendre garde aux mauvaises évaluations et à la normalisation des pgcd avant d'interpoler. En effet, si $D(X_1, \ldots, X_n)$ désigne le pgcd de P et Q et $G(X_1, \ldots, X_{n-1})$ le pgcd de $P(X_1, \ldots, X_{n-1}, \alpha)$ et de $Q(X_1, \ldots, X_{n-1}, \alpha)$, on peut seulement dire $D(X_1, \ldots, X_{n-1}, \alpha)$ divise G. Plusieurs cas sont donc possibles lorsqu'on évalue en un nouveau point α:

- l'un des degrés de G est plus petit que le degré du polynôme D' reconstruit par interpolation jusque là. Dans ce cas, toutes les évaluations qui ont conduit à reconstruire D' étaient mauvaises. Il faut recommencer l'interpolation à zéro ou à partir de G (si tous les degrés de G sont inférieurs ou égaux aux degrés du D' reconstruit).

- l'un des degrés de G est plus grand que le degré du D' reconstruit jusque là. Il faut alors ignorer α.
- Tous les degrés de G sont égaux aux degrés du D' reconstruit jusque là. Dans ce cas, G est un multiple entier du polynôme D' reconstruit jusque là et évalué en $X_n = \alpha$. Si on suppose qu'on a pu s'arranger pour que ce multiple soit 1, on ajoute le point α aux points d'évaluation précédents α_j en posant:
$$D' = D' + (G - D')\frac{\prod_{\alpha_j}(X_n - \alpha_j)}{\prod_{\alpha_j}(\alpha - \alpha_j)}$$

On voit que les mauvaises évaluations se détectent simplement par les degrés. Pour la normalisation, on utilise une petite astuce: au lieu de reconstruire le pgcd D, on va reconstruire un multiple du pgcd D (ce multiple appartiendra à $\mathbb{Z}[X_n]$). On voit maintenant P et Q comme des polynômes en $n-1$ variables $X_1, \ldots, X_{n-1}$ à coefficients dans $\mathbb{Z}[X_n]$. Alors lcoeff(D), le coefficient dominant de D (relativement à l'ordre lexicographique sur les variables $X_1, ..., X_{n-1}$), est un polynôme en X_n qui divise le coefficient dominant de P et de Q donc divise le coefficient dominant du pgcd des coefficients dominants de P et de Q. On va donc reconstruire le polynôme :

$$D' = D\frac{\Delta(X_n)}{\text{lcoeff}(D)(X_n)}, \Delta(X_n) = \text{pgcd}(\text{lcoeff}(P)(X_n), \text{lcoeff}(Q)(X_n))$$

c'est-à-dire D multiplié par un polynôme qui ne dépend que de X_n.

Revenons à G en un point α de bonne évaluation. C'est un multiple entier de $D(X_1, \ldots, X_{n-1}, \alpha)$:

$$G = \beta D(X_1, \ldots, X_{n-1}, \alpha)$$

Donc, comme polynômes de $X_1, ..., X_{n-1}$ à coefficients dans $\mathbb{Z}[X_n]$ ou dans $\mathbb{Z}$, $\text{lcoeff}(G) = \beta\,\text{lcoeff}(D)_{|X_n=\alpha}$. Comme lcoeff($D$) divise $\Delta(X_n)$, il en est de même en $X_n = \alpha$ donc lcoeff(G) divise $\beta\Delta(\alpha)$. On en déduit que $\Delta(\alpha)G$ qui est divisible par $\Delta(\alpha)\beta$ est divisible par lcoeff(G). On va donc considérer le polynôme $\Delta(\alpha)G/\text{lcoeff}(G)$: ses coefficients sont entiers et son coefficient dominant est

$$\Delta(\alpha) = \text{lcoeff}(D'(X_1, \ldots, X_{n-1}, \alpha))$$

donc

$$\Delta(\alpha)G/\text{lcoeff}(G) = D'(X_1, \ldots, X_{n-1}, \alpha)$$

Algorithme du pgcd modulaire à plusieurs variables (interpolation dense):

Arguments: 2 polynômes primitifs P et Q de n variables $X_1, \ldots, X_n$ à coefficients entiers. Renvoie le pgcd de P et Q.

1. Si $n = 1$, renvoyer le pgcd de P et Q en une variable.

2. Test rapide de pgcd trivial par rapport à X_n. On cherche des $n-1$-uplets α tels que $P(\alpha, X_n)$ et $Q(\alpha, X_n)$ soient de même degré que P et Q par rapport à la variable X_n. On calcule le pgcd G de ces 2 polynômes en une variable. Si le pgcd est constant, alors on retourne le pgcd des coefficients de P et Q.

3. On divise P et Q par leur contenu respectifs vu comme polynômes en $X_1, \ldots, X_{n-1}$ à coefficients dans $\mathbb{Z}[X_n]$, on note $C(X_n)$ le pgcd des contenus. On calcule aussi le pgcd $\Delta(X_n)$ des coefficients dominants de P et de Q.

4. On initialise D' le pgcd reconstruit à 0, $I(X_n)$ le polynôme d'interpolation à 1, $\delta = (\delta_1, ..., \delta_{n-1})$ la liste des degrés partiels du pgcd par rapport à $X_1, \ldots, X_{n-1}$ au minimum des degrés partiels de P et Q par rapport à $X_1, \ldots, X_{n-1}$, e le nombre d'évaluation à 0 et E l'ensemble des points d'interpolation à la liste vide.

5. Boucle infinie:

   - Faire α=entier aléatoire n'appartenant pas à E jusqu'à ce que

   $$\text{degre}(P(X_1, \ldots, X_{n-1}, \alpha))\text{=degre}_{X_n}(P(X_1, \ldots, X_n)$$
   $$\text{degre}(Q(X_1, \ldots, X_{n-1}, \alpha)) = \text{degre}_{X_n}(Q(X_1, \ldots, X_n))$$

   - Calculer le pgcd $G(X_1, \ldots, X_{n-1})$ en $n-1$ variables de $P(X_1, \ldots, X_{n-1}, \alpha)$ et $Q(X_1, \ldots, X_{n-1}, \alpha)$.
   - Si $\text{degre}(G)_i < \delta_i$ pour un indice au moins. Si $\text{degre}(G) \leqslant \delta$, on pose $\delta = \text{degre}(G)$, $D' = G\frac{\Delta(\alpha)}{\text{lcoeff}(G)}$, $I = X_n - \alpha$, $e = 1$ et $E = [\alpha]$, sinon on pose $\delta = \min(\delta, \text{degre}(G)), D' = 0, I = 1, e = 0, E = []$. On passe à l'itération suivante.
   - Si $\text{degre}(G) > \delta$, on passe à l'itération suivante.
   - Si $\text{degre}(G) = \delta$, on interpole:
     - $\rightarrow$ $G := G\frac{\Delta(\alpha)}{\text{lcoeff}(G)}$
     - $\rightarrow$ $D' := D' + \frac{I(X_n)}{\prod_{\alpha_j \in E}(\alpha - \alpha_j)}(G - D'(X_1, \ldots, X_{n-1}, \alpha))$
     - $\rightarrow$ $I := I * (X_n - \alpha)$
     - $\rightarrow$ $e := e + 1$ et ajouter α à E
     - $\rightarrow$ Si e est strictement plus grand que le minimum des degrés partiels de P et Q par rapport à X_n, on pose $\tilde{D}$ la partie primitive de D'(vu comme polynôme à coefficients dans $\mathbb{Z}[X_n]$), on teste si P et Q sont divisibles par $\tilde{D}$, si c'est le cas, on renvoie $D = C(X_n)\tilde{D}$

On observe que dans cet algorithme, on fait le test de divisibilite de $\tilde{D}$ par P et Q. En effet, même après avoir évalué en suffisamment de points, rien n'indique que tous ces points sont des points de bonne évaluation. En pratique cela reste extrêmement improbable. En pratique, on teste la divisibilité plus tôt, dès que D' n'est pas modifié par l'ajout d'un nouveau point à la liste des α_j.

Il existe une variation de cet algorithme, appelé SPMOD (sparse modular), qui suppose que seuls les coefficients non nuls du pgcd en $n-1$ variables sont encore non nuls en n variables (ce qui a de fortes chances d'être le cas). L'étape d'interpolation est alors remplacée par la résolution d'un sous-système d'un système de Vandermonde. Cette variation est intéressante si le nombre de coefficients non nuls en $n-1$ variables est petit devant le degré. Si elle échoue, on revient à l'interpolation dense.

Notons enfin qu'on peut appliquer cette méthode lorsque les coefficients de P et Q sont dans $\mathbb{Z}/n\mathbb{Z}$ mais il faut alors vérifier qu'on dispose de suffisamment de points d'interpolation. Ce qui en combinant avec l'algorithme modulaire à une variable donne un algorithme doublement modulaire pour calculer le pgcd de 2 polynômes à coefficients entiers. C'est cette méthode qu'utilise par exemple MuPAD (en essayant d'abord SPMOD puis l'interpolation dense).

Exemple:

Dans cet exemple, on donne F et G sous forme factorisée, le but étant de faire comprendre l'algorithme. En utilisation normale, on n'exécuterait cet algorithme que si F et G étaient développés.

$P=((x+1)y+x^2+1)(y^2+xy+1), Q=((x+1)y+x^2+1)(y^2-xy-1)$.

Prenons x comme variable X_1 et y comme variable X_2. Les coefficients dominants de P et Q sont respectivement y et $-y$ donc $\Delta = y$.

En $y=0$, $P(x,0)=x^2+1$ n'est pas du bon degré.

En $y=1$, $P(x,1)=(x+x^2+2)(x+2)$ et $Q(x,1)=(x+x^2+2)(-x)$ sont du bon degré. Leur pgcd est $G=x^2+x+2$, $\Delta(1)=1$, donc $D'=x^2+x+1$. On teste la divisibilité de P par D', le teste échoue.

En $y=2$, $P(x,2)=(x^2+2x+3)(2x+5)$ et $Q(x,2)=(x^2+2x+3)(-2x+3)$ donc $G=x^2+2x+3$, $\Delta(2)=2$. On interpole:

$$D'=x^2+x+2+\frac{y-1}{2-1}(2(x^2+2x+3)-(x^2+x+2))=y(x^2+3x+4)-(2x+2)$$

On teste la divisibilité de P par D', le test échoue.

En $y=3$, $P(x,3)=(x^2+3x+4)(3x+10)$ et $Q(x,3)=(x^2+3x+4)(-3x+8)$ donc $G=x^2+3x+4$, $\Delta(3)=3$. On interpole:

$$\begin{aligned} D' &= y(x^2+3x+4)-(2x+2)+ \\ & \frac{(y-2)(y-1)}{(3-2)(3-1)}\left(3(x^2+3x+4)-(3(x^2+3x+4)-(2x+2))\right) \end{aligned}$$

donc

$$D'=y(x^2+3x+4)-(2x+2)+\frac{(y-2)(y-1)}{2}(-2x-2)=x^2y+xy^2+y^2+y$$

On divise D' par son contenu et on trouve $x^2 + xy + y + 1$ qui est bien le pgcd de P et Q.

3.3 EZGCD.

Il s'agit d'une méthode p-adique. On évalue toutes les variables sauf une, on calcule le pgcd en une variable et on remonte au pgcd variable par variable (EEZGCD) ou toutes les variables simultanément (EZGCD) par un lemme de Hensel. Il semble qu'il est plus efficace de remonter les variables séparément.

Soit donc F et G deux polynômes primitifs dépendant des variables $X_1, \dots, X_n$ de pgcd D, on fixe une des variables qu'on appelera X_1 dans la suite. Soient $\mathrm{lcoeff}(F)$ et $\mathrm{lcoeff}(G)$ les coefficients dominants de F et G par rapport à X_1. On évalue F et G en un $n-1$ uplet b tel que le degré de F et G par rapport à X_1 soit conservé après evaluation en b. On suppose que $D_b(X_1) = \mathrm{pgcd}(F(b), G(b))$ a le même degré que $D(b)$. On a donc l'égalité:

$$(F * \mathrm{lcoeff}(F))(b) = \left(D_b \frac{\mathrm{lcoeff}(F(b))}{\mathrm{lcoeff}(D_b)} \right) * \left(\frac{F(b)}{D_b} \frac{\mathrm{lcoeff}(F)(b)}{\mathrm{lcoeff}(\frac{F(b)}{D_b})} \right)$$

et de même en remplaçant F par G.

Pour pouvoir lifter cette égalité (c'est-à-dire généraliser à plusieurs variables), il faut que D_b et $\frac{F(b)}{D_b}$ soient premiers entre eux. Sinon, on peut essayer de lifter l'égalité analogue avec G. En général, on montre qu'il existe un entier j tel que D_b et $\frac{F(b)+jG(b)}{D_b}$ soient premiers entre eux. En effet, sinon au moins un des facteurs irréductibles de D_b va diviser $\frac{F(b)+jG(b)}{D_b}$ pour deux valeurs distinctes de j et va donc diviser à la fois $\frac{F(b)}{D_b}$ et $\frac{G(b)}{D_b}$ en contradiction avec la définition de $D_b = \mathrm{pgcd}(F(b), G(b))$. On lifte alors l'égalité obtenue en remplaçant F par $(F + kG)$ ci-dessus. Dans la suite, on suppose qu'on peut prendre $j = 0$ pour alléger les notations.

On va aussi supposer que $b = 0$. Sinon, on fait un changement d'origine sur les polynômes F et G pour que $b = 0$ convienne, on calcule le pgcd et on lui applique la translation d'origine opposée.

On adopte ensuite la notation suivante: si k est un entier, on dit qu'un polynôme P est un $O(k)$ si la valuation de P vu comme polynôme en $X_2, \dots, X_n$ à coefficients dans $\mathbb{Z}[X_1]$ est supérieure ou égale à k, ou de manière équivalente si

$$P(X_1, hX_2, \dots, hX_n) = O_{h \to 0}(h^k)$$

L'égalité à lifter se réécrit donc:

$$F\, \mathrm{lcoeff}(F) = P_0 Q_0 + O(1)$$

où $P_0 = D_b \frac{\mathrm{lcoeff}(F(b))}{\mathrm{lcoeff}(D_b)}$ et $Q_0 = \frac{F(b)}{D_b} \frac{\mathrm{lcoeff}(F)(b)}{\mathrm{lcoeff}(\frac{F(b)}{D_b})}$ sont premiers entre eux et de degré 0 par rapport aux variables $X_2, \dots, X_n$. Cherchons $P_1 = O(1)$ et $Q_1 = O(1)$ de degré 1 par rapport aux variables $X_2, \dots, X_n$ tels que

$$F\, \mathrm{lcoeff}(F) = (P_0 + P_1)(Q_0 + Q_1) + O(2)$$

Il faut donc résoudre

$$F \operatorname{lcoeff}(F) - P_0 Q_0 = P_0 Q_1 + Q_0 P_1 + O(2)$$

On peut alors appliquer l'identité de Bézout qui permet de déterminer des polynômes P_1 et Q_1 satisfaisant l'égalité ci-dessus (avec comme reste $O(2)$ nul) puisque P_0 et Q_0 sont premiers entre eux. De plus, on choisit P_1 et Q_1 tels que $\text{degre}_{X_1} P_1 \leqslant \text{degre}_{X_1}(F) - \text{degre}(Q_0) = \text{degre}(P_0)$ et $\text{degre}_{X_1}(Q_1) \leqslant \text{degre}(Q_0)$ et $\operatorname{lcoeff}_{X_1}(P_0 + P_1) + O(2) = \operatorname{lcoeff}_{X_1}(Q_0 + Q_1) + O(2) = \operatorname{lcoeff}_{X_1}(F)$. On tronque ensuite P_1 et Q_1 en ne conservant que les termes de degré 1 par rapport à $X_2, \ldots, X_n$.

On trouve de la même manière par récurrence P_k et Q_k homogènes de degré k par rapport à $X_2, \ldots, X_k$, de degré par rapport à X_1 respectivement inférieur aux degrés de Q_0 et de P_0 et tels que

$$F \operatorname{lcoeff}(F) = (P_0 + \ldots. + P_k)(Q_0 + \ldots. + Q_k) + O(k+1) \qquad (6)$$

et $\operatorname{lcoeff}(F) = \operatorname{lcoeff}_{X_1}(P_0 + \ldots. + P_k) + O(k+1) = \operatorname{lcoeff}_{X_1}(Q_0 + \ldots. + Q_k) + O(k+1)$.

Si on est bien en un point de bonne évaluation et si k est plus grand que le degré total (par rapport aux variables $X_2, \ldots, X_n$) du polynôme $F \operatorname{lcoeff}(F)$ on va vérifier que $P_0 + \ldots. + P_k = D \frac{\operatorname{lcoeff}(F)}{\operatorname{lcoeff}(D)}$. En effet, si on a deux suites de polynômes P et P' et Q et Q' satisfaisant (6) avec les même termes de degré zéro P_0 et Q_0, alors en prenant la différence, on obtient:

$$(P_0 + P_1 \ldots + P_k)(Q_0 + Q_1 \ldots + Q_k) = (P_0 + P'_1 \ldots + P'_k)(Q_0 + Q'_1 \ldots + Q'_k) + O(k+1)$$

On égale alors les termes homogènes de degré j, pour $j = 1$, on obtient $P_0(Q_1 - Q'_1) = Q_0(P_1 - P'_1)$, donc Q_0 divise $Q_1 - Q'_1$ qui est de degré strictement inférieur au degré de Q_0 par rapport à X_1 (car on a l'inégalité large et les termes de plus haut degré sont égaux), donc $Q_1 = Q'_1$ et $P_1 = P'_1$. On montre de la même manière que $Q_j = Q'_j$ et $P_j = P'_j$. L'écriture est donc unique, c'est donc l'écriture en polynôme homogène de degré croissant de $D \frac{\operatorname{lcoeff}(F)}{\operatorname{lcoeff}(D)}$ que l'on reconstruit.

Cet algorithme permet donc de reconstruire D, il suffit de tester à chaque étape si $P_0 + \ldots. + P_k$ divise $F \operatorname{lcoeff}(F)$. On appelle cette méthode de remontée lemme de Hensel linéaire. Il existe une variante dite lemme de Hensel quadratique qui consiste à passer de $O(k)$ à $O(2k)$. Elle nécessite toutefois un calcul supplémentaire, celui de l'identité de Bézout à $O(2k)$ près pour les polynômes $P_0 + \ldots. + P_{k-1}$ et $Q_0 + \ldots. + Q_{k-1}$. Ce calcul se fait également par lifting.

Algorithme EZGCD (Hensel linéaire)

Arguments: 2 polynômes F et G à coefficients entiers et primitifs. Renvoie le pgcd de F et G ou false.

1. Evaluer F et G en $(X_2, \ldots, X_n) = (0, \ldots, 0)$, vérifier que les coefficients dominants de F et de G ne s'annulent pas. Calculer le pgcd D_b de $F(0)$ et de $G(0)$. Prendre un autre point d'évaluation au hasard qui n'annule pas

les coefficients dominants de F et de G et vérifier que le pgcd a le même degré que D_b. Sinon, renvoyer false (on peut aussi faire une translation d'origine de F et de G en un autre point mais cela diminue l'efficacité de l'algorithme).

2. On note lcF et lcG les coefficients dominants de F et de G par rapport à X_1.

3. Si $\text{degre}(F) \leqslant \text{degre}(G)$ et $\text{degre}(D_b) = \text{degre}(G)$ et F divise G renvoyer F

4. Si $\text{degre}(G) < \text{degre}(F)$ et $\text{degre}(D_b) = \text{degre}(F)$ et G divise F renvoyer G

5. Si $\text{degre}(F) = \text{degre}(D_b)$ ou si $\text{degre}(G) = \text{degre}(D_b)$ renvoyer false

6. Boucle infinie sur j entier initialisé à 0, incrémenté de 1 à chaque itération: si $\text{pgcd}(D_b, \frac{F(0)+jG(0)}{D_b}) = C$ constant, alors arrêter la boucle

7. Lifter l'égalité $(F + jG)(\text{lcF} + j\,\text{lcG})(0) = \left(D_b \frac{(\text{lcF} + j\,\text{lcG})(0)}{\text{lcoeff}(D_b)}\right) * \ldots$. par remontée de Hensel linéaire ou quadratique. Si le résultat est false, renvoyer false. Sinon renvoyer le premier polynôme du résultat divisé par son contenu vu comme polynôme en X_1 à coefficients dans $\mathbb{Z}[X_2, \ldots, X_n]$.

Remontée de Hensel linéaire:

Arguments: F un polynôme, lcF=lcoeff(F) son coefficient dominant, P_0 un facteur de $F(0)$ ayant comme coefficient dominant lcF(0) et dont le cofacteur Q_0 est premier avec P_0.

Renvoie deux polynômes P et Q tels que $F\,\text{lcF} = PQ$ et $P(0) = P_0$ et $\text{lcoeff}(P) = \text{lcoeff}(Q) = \text{lcF}$.

1. Soit $G = F\,\text{lcF}$, , $Q_0 = G(0)/P_0$, $P = P_0$, $Q = Q_0$.

2. Déterminer les deux polynômes U et V de l'identité de Bézout (tels que $P_0U + Q_0V = d$ où d est un entier).

3. Boucle infinie avec un compteur k initialisé à 1, incrémenté de 1 à chaque itération

   - Si $k > \text{degre}_{X_2,\ldots,X_n}(G)$, renvoyer false.
   - Si P divise G, renvoyer P et G/P.
   - Soit $H = G - PQ = O(k)$. Soit $u = U\frac{H}{d}$ et $v = V\frac{H}{d}$, on a $P_0u + Q_0v = H$
   - Remplacer v par le reste de la division euclidienne de v par P_0 et u par le reste de la division euclidienne de u par Q_0. La somme des deux quotients est égale au quotient euclidien de H par P_0Q_0, c'est-à-dire

au coefficient dominant de H divisé par le produit des coefficients dominants de P_0 et Q_0 (qui sont égaux) donc on a l'égalité:

$$P_0 u + Q_0 v = H - \frac{\operatorname{lcoeff}(H)}{\operatorname{lcoeff}(P_0)^2} P_0 Q_0$$

- Soit $\alpha = (\operatorname{lcoeff}(F) - \operatorname{lcoeff}(P))/\operatorname{lcoeff}(P_0)$ et $\beta = (\operatorname{lcoeff}(F) - \operatorname{lcoeff}(Q))/\operatorname{lcoeff}(P_0)$. On ajoute αP_0 à v, ainsi $\operatorname{lcoeff}(P+v) = \operatorname{lcoeff}(F) + O(k+1)$ et βQ_0 à u, ainsi $\operatorname{lcoeff}(Q+u) = \operatorname{lcoeff}(F) + O(k+1)$

  Remarque: on montre alors que $\alpha + \beta = \frac{\operatorname{lcoeff}(H)}{\operatorname{lcoeff}(P_0 Q_0)} + O(k+1)$ donc $P_0 u + Q_0 v = H + O(k+1)$ en utilisant les propriétés :

  $\operatorname{lcoeff}(F) = \operatorname{lcoeff}(P) + O(k) = \operatorname{lcoeff}(Q) + O(k) = \operatorname{lcoeff}(P_0) + O(1)$

- Réduire u et v en éliminant les termes de degré strictement supérieur à k par rapport à $X_2, \ldots, X_n$. S'il reste un coefficient non entier, renvoyer false
- Remplacer P par $P + v$ et Q par $Q + u$, passer à l'itération suivante.

Exemple:

$F = ((x+1)y + x^2 + 1)(y^2 + xy + 1), G = ((x+1)y + x^2 + 1)(y^2 - xy - 1)$

On a $F(0, y) = (y+1)(y^2+1)$ et $G(0, y) = (y+1)(y^2-1)$, le pgcd est donc $D_b = (y+1)$. On remarque que D_b est premier avec le cofacteur de F mais pas avec le cofacteur de G. Si on évalue en un autre point, par exemple $x = 1$, on trouve un pgcd D_1 de même degré, donc 0 est vraissemblablement un bon point d'évaluation (ici on en est sûr puisque le pgcd de F et G se calcule à vue...). On a $\operatorname{lcoeff}(F) = x+1$, on va donc lifter $G = ((x+1)y+x^2+1)(y^2+xy+1)(x+1) = PQ$ où $P_0 = (y+1)$ et $Q_0 = (y^2+1)$.

On calcule les polynômes de l'identité de Bézout $U = (1-y)$ et $V = 1$ avec $d = 2$, puis à l'ordre $k = 1$:

$$H = G - P_0 Q_0 = (2y^3 + 2y^2 + 3y + 1)x + O(2)$$

donc $u = \operatorname{reste}(UH/d, Q_0) = xy$ et $v = \operatorname{reste}(VH/d, P_0) = -x$.

Donc $Q_1 = xy + \alpha Q_0$ avec $\alpha = (x+1-1)/\operatorname{lcoeff}(P_0) = x$ et $Q_0 + Q_1 = (y^2+1)(x+1)+xy$. De même, $P_1 = -x + \beta P_0$, avec $\beta = (x+1-1)/\operatorname{lcoeff}(P_0) = x$ donc $P_0 + P_1 = (y+1)(x+1) - x$. On remarque que $P_0 + P_1$ et $Q_0 + Q_1$ sont bien à $O(2)$ près les facteurs de $F \operatorname{lcoeff}(F)$:

$$P = (x+1)y+x^2+1 = P_0+P_1+O(2),\ Q = (x+1)(y^2+xy+1) = Q_0+Q_1+O(2)$$

Une deuxième itération est nécessaire. On calcule

$$H = G - (P_0 + P_1)(Q_0 + Q_1) = (2y^2 + y + 1)x^2 + O(3)$$

puis $\operatorname{reste}(UH/d, Q_0) = yx^2$ et $\operatorname{reste}(VH/d, P_0) = x^2$. Ici les coefficients α et β sont nuls car $\operatorname{lcoeff}(F)$ n'a pas de partie homogène de degré 2. On trouve alors

$P = P_0 + P_1 + P_2$ et $Q = Q_0 + Q_1 + Q_2$. Pour calculer le pgcd, il suffit de calculer la partie primitive de P vu comme polynôme en y, ici c'est encore P car le contenu de P est 1 (remarque: pour Q le contenu est $x + 1$).
On trouve donc P comme pgcd.

4 Quel algorithme choisir?

Il est toujours judicieux de faire une évaluation en quelques $n - 1$ uplets pour traquer les pgcd triviaux. (E)EZGCD sera efficace si (0,...,0) est un point de bonne évaluation et si le nombre de remontées nécessaires pour le lemme de Hensel est petit donc pour les pgcd de petit degré, GCDMOD est aussi efficace si le degré du pgcd est petit. Le sous-résultant est efficace pour les pgcd de grand degré car il y a alors peu de divisions euclidiennes à effectuer et les coefficients n'ont pas trop le temps de croitre. SPMOD est intéressant pour les polynômes creux de pgcd non trivial creux. GCDHEU est intéressant pour les problèmes relativement petits.

Avec des machines multiprocesseurs, on a probablement intérêt à lancer en parallèle plusieurs algorithmes et à s'arrêter dès que l'un deux recontre le succès.

5 Pour en savoir plus.

Parmi les références citées dans le premier article, ce sont les livres de Knuth, H. Cohen, et Davenport-Siret-Tournier qui traitent des algorithmes de pgcd. On peut bien sûr consulter le source de son système de calcul formel lorsqu'il est disponible :

- pour MuPAD sur un système Unix, depuis le répertoire d'installation de MuPAD (en général `/usr/local/MuPAD`) après avoir désarchivé le fichier `lib.tar` du répertoire `share/lib` par la commande
  `cd share/lib && tar xvf lib.tar`
  on trouve les algorithmes de calcul de PGCD dans le répertoire
  `share/lib/lib/POLYLIB/GCD`
- Pour l'algorithme EZGCD, je me suis inspiré de l'implémentation de Singular (logiciel libre disponible à `www.singular.uni-kl.de`)

Sur le web on trouve quelques articles en lignes sur le sujet en cherchant les mots clefs GCDHEU, EZGCD, SPMOD sur un moteur de recherche, il y a par exemple une description un peu différente du pgcd heuristique sur:
`www.inf.ethz.ch/personal/gonnet/CAII/HeuristicAlgorithms/node1.html`
et un article de comparaison de ces algorithmes par Fateman et Liao (dont la référence bibliographique est Evaluation of the heuristic polynomial GCD. in: ISSAC pages 240–247, 1995). Quelques autres références :

- K.O.Geddes et al "Alg. for Computer Algebra", Kluwer 1992.

- pour GCDHEU Char, Geddes, Gonnet, Gcdheu: Heuristic polynomial gcd algorithm based on integer gcd computation, in: Journal of Symbolic Computation, 7:31–48, 1989.
- pour SPMOD "Probabilistic Algorithms for Sparse Polynomials", in: Symbolic & Algebraic Comp. (Ed E.W.Ng), Springer 1979, pp216,