CALCUL FORMEL : CE QU'IL Y A DANS LA BOÎTE

par

Paul Zimmermann

Introduction

Le calcul formel est aujourd'hui en plein essor. Pour preuve, on dénombre à l'heure actuelle pas moins de sept systèmes généraux (Axiom, Derive, Macsyma, Maple, Mathematica, MuPAD, Reduce), une multitude de systèmes spécialisés (GAP, GB, Macaulay, Magma, Pari/GP, ...), et une abondante littérature, y compris en français [**5, 7, 11, 14**]. Cet essor s'explique d'une part par l'accessibilité de ces systèmes (ils fonctionnent pour la plupart sur un Mac ou un PC), leur apprentissage aisé, lié principalement au fait que ce sont des langages interprétés (et non compilés comme Pascal, C ou Fortran) disposant d'une interface agréable et d'une bonne aide en ligne, d'autre part par l'effort d'enseignement qui est fait, notamment en France avec l'introduction du calcul formel dans les classes préparatoires aux grandes écoles.

Si les systèmes de calcul formel commencent à être bien connus, du moins dans le monde universitaire (il faudra encore quelques années pour que les formations actuelles portent leurs fruits dans l'industrie), on sait en général moins bien comment ces systèmes fonctionnent. Comment fait Axiom pour factoriser le polynôme x^5+x+1 ? Quelle méthode emploie Macsyma pour factoriser l'entier $2^{64}+1$? Comment Mathematica calcule-t-il $\int_0^2 \sqrt{x+1/x-2}\,dx$? Quel principe utilise Maple pour déterminer la valeur de $\sum_{k=0}^{n} \binom{n}{k}^2$? Comment MuPAD résout-il la récurrence $nu_{n+2} = 5u_{n+1} + (n+1)u_n$? La méconnaissance des méthodes utilisées entraîne de fait celle des classes de problèmes que les systèmes savent résoudre. La conséquence est que les systèmes de calcul formel sont un peu considérés comme des « boîtes magiques », et que l'on n'a aucune idée a priori de la difficulté intrinsèque d'un problème donné pour les systèmes de calcul formel. Lorsque le système n'arrive pas à résoudre le problème donné, est-ce parce qu'il n'est pas assez puissant ou parce qu'il n'y a pas de réponse ? Certaines méthodes utilisées par les systèmes sont des algorithmes de *décision*, c'est-à-dire que l'absence de réponse positive signifie « il n'existe pas de solution dans la classe considérée ». C'est le cas notamment de l'intégration indéfinie des fonctions élémentaires (algorithme de Risch) et de la sommation indéfinie des fonctions hypergéométriques (algorithme de Gosper). La connaissance des méthodes utilisées permet alors de déduire plus d'information que le système n'en donne apparemment.

À ce jour, aucun ouvrage en français ne décrit l'ensemble des algorithmes utilisés en calcul formel. Dans [**11**], nous avons essayé de décrire les classes de problèmes que les systèmes savent traiter, mais sans trop rentrer dans les détails car nous ne voulions pas introduire trop de concepts mathématiques. En anglais, il existe un ouvrage général mais malheureusement cher [**10**], et deux ouvrages spécialisés que nous recommandons, le premier sur la sommation et la preuve d'identités discrètes [**15**], le second sur l'intégration [**3**]. Citons également le livre en préparation *Modern Computer Algebra*, par Joachim von zur Gathen et Jürgen Gerhard,

Ce document, issu de notes de cours de DEA *Quelques algorithmes de calcul symbolique* donné à Nancy 1, indique de manière succincte le fonctionnement de cinq algorithmes fondamentaux utilisés par les systèmes de calcul formel : l'algorithme de Gosper pour le calcul de sommes indéfinies $\sum_n f_n$, l'algorithme de Zeilberger pour le calcul de sommes définies $\sum_{k=0}^{n} f_{n,k}$, l'algorithme de Berlekamp pour la factorisation de polynômes à coefficients dans un corps fini, l'algorithme de Zassenhaus pour la factorisation de polynômes à coefficients entiers, et l'algorithme de Lenstra pour la factorisation d'entiers. Chaque section peut être lue indépendamment des autres, sauf l'algorithme de Zeilberger qui utilise celui de Gosper, et l'algorithme de Zassenhaus qui recourt à celui de Berlekamp. Des références bibliographiques sont fournies, où l'on pourra trouver des informations complémentaires sur ces algorithmes, notamment leur complexité, qui n'est pas abordée ici.

1. L'algorithme de Gosper pour le calcul de $\sum_n f_n$

On trouvera une description plus détaillée de cet algorithme dans [**15**], avec notamment une preuve des théorèmes.

***Définition 1* (Fonction hypergéométrique).** Une suite f_n est dite *hypergéométrique* (en n) ssi le rapport f_{n+1}/f_n est rationnel (en n).

***Exemple*.** 2^n et $\binom{m}{n}$ sont hypergéométriques en n (et m), mais pas $\sin n$.

***Définition 2* (Somme indéfinie).** La somme indéfinie de f_n par rapport à n, notée $\sum_n f_n$, est une expression g_n telle que $f_n = g_{n+1} - g_n$.

***Remarque*.** Comme pour l'intégration indéfinie, g_n est définie à une constante près. On peut aussi choisir la définition $f_n = g_n - g_{n-1}$, qui mène à des calculs différents.

Exemple. On a $\sum_n n = n(n-1)/2$ et $\sum_n n^2 = n(n-1)(2n-1)/6$.

En 1978, Gosper a mis au point un algorithme permettant de décider si une fonction hypergéométrique admet ou non une somme (indéfinie) hypergéométrique [**12**].

Algorithme de Gosper**.**
Entrée : une suite hypergéométrique f_n.
Sortie : une suite hypergéométrique g_n telle que $g_n = \sum_n f_n$ ou bien `fail`.

1. Calculer $r(n) = f_{n+1}/f_n$ (fraction rationnelle en n).
2. Mettre $r(n)$ sous la forme $\dfrac{a(n)}{b(n)}\dfrac{c(n+1)}{c(n)}$ où $a(n), b(n), c(n)$ sont des polynômes en n tels que si α est racine de $a(n)$, alors $\alpha + h$ n'est pas racine de $b(n)$, i.e.

(1) $$\mathrm{pgcd}(a(n), b(n+h)) = 1 \text{ pour tout entier } h \geq 0.$$

3. Chercher un polynôme $x(n)$ solution de l'équation

(2) $$a(n)x(n+1) - b(n-1)x(n) = c(n).$$

4. Si un tel $x(n)$ existe, renvoyer $g_n = \dfrac{b(n-1)x(n)}{c(n)} f_n$, sinon renvoyer `fail`.

Exemple. Soit $f_n = (4n+1)\dfrac{n!}{(2n+1)!}$. Alors $r(n) = \dfrac{4n+5}{2(2n+3)(4n+1)}$. Les polynômes $a(n) = 1, b(n) = 2(2n+3), c(n) = 4n+1$ satisfont la condition 2, et l'équation (2) devient $x(n+1) - 2(2n+1)x(n) = 4n+1$, dont une solution évidente est $x(n) = -1$. Donc $\sum_n f_n = -2n!/(2n)!$.

1.1. Mise sous la forme $\dfrac{a(n)}{b(n)}\dfrac{c(n+1)}{c(n)}$.

Soit $r(n) = Zt(n)/u(n)$ où Z est une constante, et $t(n), u(n)$ sont des polynômes premiers entre eux et unitaires (de coefficient dominant 1). On calcule tout d'abord $R(h) = \mathrm{res}_n(t(n), u(n+h))$, puis $S = \{h_1, \ldots, h_N\}$, $h_1 \leq \cdots \leq h_N$, l'ensemble des racines entières positives ou nulles de $R(h)$.

$p_0(n) \leftarrow t(n), q_0(n) \leftarrow u(n)$
`for` $j = 1$ `to` N `do`
  $s_j(n) \leftarrow \mathrm{pgcd}(p_{j-1}(n), q_{j-1}(n+h_j))$
  $p_j(n) \leftarrow p_{j-1}(n)/s_j(n)$
  $q_j(n) \leftarrow q_{j-1}(n)/s_j(n-h_j)$
$a(n) \leftarrow Zp_N(n)$
$b(n) \leftarrow q_N(n)$
$c(n) \leftarrow \prod_{i=1}^{N}\prod_{j=1}^{h_i} s_i(n-j)$

Théorème 1. *Soit K un corps de caractéristique zéro, et $r \in K[n]$ une fraction rationnelle non nulle. Alors il existe un unique triplet a, b, c de polynômes de $K[n]$ vérifiant* (1) *tels que b, c sont unitaires, $r(n) = \frac{a(n)}{b(n)} \frac{c(n+1)}{c(n)}$,* $\operatorname{pgcd}(a(n), c(n)) = 1$ *et* $\operatorname{pgcd}(b(n), c(n+1)) = 1$.

Les deux dernières conditions assurent l'unicité, qui n'est pas vraiment requise pour la bonne marche de l'algorithme. Si on prend $r(n) = \frac{n+3}{n(n+1)}$, alors

$$a(n) = 1, \quad b(n) = n, \quad c(n) = (n+1)(n+2)$$

et

$$a(n) = n - 1, \quad b(n) = n^2, \quad c(n) = (n+1)(n+2)(n-1)$$

vérifient (1) et

$$r(n) = \frac{a(n)}{b(n)} \frac{c(n+1)}{c(n)},$$

mais le second triplet ne vérifie pas $\operatorname{pgcd}(a(n), c(n)) = 1$.

1.2. Recherche des solutions $x(n)$ de l'équation de Gosper

Théorème 2 **([12])**. *Soient $a(n), b(n), c(n)$ des polynômes en n vérifiant* (1). *Alors toute fraction rationnelle $x(n)$ solution de* (2) *est nécessairement un polynôme.*

Pour déterminer le degré d d'une éventuelle solution $x(n)$ de (2), on distingue deux cas suivant les degrés et coefficients dominants de $a(n)$ et $b(n)$.

Cas 1 : $\deg a(n) \neq \deg b(n)$ *ou* $\operatorname{lc} a(n) \neq \operatorname{lc} b(n)$. Dans ce cas le degré du membre gauche de (2) est $d + \max\{\deg a(n), \deg b(n)\}$, et donc on a :

$$d = \deg c(n) - \max\{\deg a(n), \deg b(n)\}.$$

Cas 2 : $\deg a(n) = \deg b(n)$ *et* $\operatorname{lc} a(n) = \operatorname{lc} b(n) = \lambda$. Les termes dominants de (2) s'annulent. On distingue à nouveau deux sous-cas (qui ne sont pas incompatibles, par conséquent il faut prendre le plus grand degré obtenu, cf $\frac{1}{n(n+1)}$ où 2a donne $d = 0$ et 2b donne $d = 1$) :

(2a) Si les termes de second ordre du membre gauche de (2) ne s'annulent pas, alors :

$$d = \deg c(n) - \deg a(n) + 1.$$

(2b) Si les termes de second ordre du membre gauche de (2) s'annulent, soient $a(n) = \lambda n^k + A n^{k-1} + \cdots$, $b(n-1) = \lambda n^k + B n^{k-1} + \cdots$, alors nécessairement $d = \frac{B - A}{\lambda}$.

Voici deux exemples avec MuPAD 1.3 : le premier (cas 1) indique que la somme indéfinie de $(4n+1)\dfrac{n!}{(2n+1)!}$ est $-2n!/(2n)!$ (après simplification), le second (cas 1) que $n!$ n'a pas de somme indéfinie hypergéométrique.

```
>> sum((4*n+1)*n!/(2*n+1)!, n);

                         - 2 fact(n) - 4 n fact(n)
                         -------------------------
                               fact(2 n + 1)

>> sum(n!, n);

                              sum(fact(n), n)
```

2. L'algorithme de Zeilberger pour le calcul de $\sum_{k=0}^{n} F(n,k)$

L'algorithme de Zeilberger [19] permet de calculer des sommes définies

$$f(n)=\sum_{k=0}^{n} F(n,k)$$

même lorsque l'expression $F(n,k)$ n'admet pas de somme indéfinie $\sum_k F(n,k)$, i.e. lorsque l'algorithme de Gosper échoue. C'est un peu l'analogue de la méthode de Laplace pour calculer des intégrales définies. L'exemple type est $F(n,k)=\binom{n}{k}x^k$:

```
>> sum(binomial(n,k)*x^k, k), sum(binomial(n,k)*x^k, k=0..n);

                         k                               n
                  sum(x  binomial(n, k), k), (x + 1)
```

L'idée de Zeilberger est d'essayer de trouver une formule de récurrence pour $f(n)$ en fonction de n, que l'on peut ensuite résoudre pour trouver une forme close. On peut aussi utiliser directement la récurrence et le calcul des premiers termes pour montrer des identités du genre :

$$\sum_{s=0}^{2m}(-1)^s\binom{2m}{s}^3=(-1)^m\frac{(3m)!}{(m!)^3}.$$

```
>> sum((-1)^s*binomial(2*m,s)^3,s=0..2*m);

     /    /                3 u1(m) (3 m + 1) (3 m + 2)                      \ \
     |    | u1(m + 1) = - ---------------------------, u1(m), {u1(0) = 1} | |
solve| rec|                                2                                | |
     \    \                         (m + 1)                                 / /
```

L'astuce de génie de Zeilberger est d'utiliser une version étendue de l'algorithme de Gosper (qui normalement calcule des sommes indéfinies) pour justement calculer des sommes *définies*, en ajoutant des paramètres qui seront déterminés au cours de l'algorithme.

2.1. Fonctions proprement hypergéométriques

Définition 3 **([15**, p. 64]). Une fonction $F(n,k)$ est *proprement hypergéométrique* si elle peut se mettre sous la forme

$$F(n,k) = P(n,k)\frac{\prod_{i=1}^{u}(a_i n + b_i k + c_i)!}{\prod_{i=1}^{v}(a'_i n + b'_i k + c'_i)!}\, x^k$$

où P est un polynôme, les a_i, b_i, a'_i, b'_i sont des entiers, et u, v sont des entiers positifs ou nuls.

Exemple. L'expression $\binom{n}{k}2^k = \frac{n!}{k!(n-k)!}2^k$ est proprement hypergéométrique, ainsi que $\frac{1}{n+3k+1} = \frac{(n+3k)!}{(n+3k+1)!}$, mais pas $\frac{1}{n^2+k^2+1}$.

Théorème 3 **(Wilf, Zeilberger)**. *Soit $F(n,k)$ une fonction proprement hypergéométrique. Alors F satisfait une récurrence de la forme*

$$\sum_{j=0}^{J} a_j(n)F(n+j,k) = G(n,k+1) - G(n,k) \tag{3}$$

où de plus $G(n,k)/F(n,k)$ est une fraction rationnelle en n et k.

Non seulement une telle récurrence existe donc pour toute fonction proprement hypergéométrique, mais en plus Wilf et Zeilberger ont prouvé une borne sur l'ordre J : $J \le |b_1| + \cdots + |b_u| + |b'_1| + \cdots + |b'_v|$ où les b_i, b'_i sont ceux de la définition 3.

Algorithme de Zeilberger.
Entrée : une expression $F(n,k)$ hypergéométrique en n et k et un entier J.
Sortie : une récurrence linéaire d'ordre au plus J pour $f(n) = \sum_{k=0}^{n} F(n,k)$ ou bien `fail`.
1. Calculer $r_{n,k} = F(n,k)/F(n-1,k)$.
2. Calculer les fractions rationnelles $R_0 = 1$ et $R_j = R_{j-1}r_{n+j,k}$ pour $1 \le j \le J$.
3. Appliquer l'algorithme de Gosper étendu (cf ci-dessous) par rapport à k à $(R_0 + C_1R_1 + \cdots + C_JR_J)F(n,k)$ où les C_j sont des indéterminées ne dépendant pas de k.
4. Si Gosper retourne `fail`, faire de même. Sinon, soit $G(n,k)$ la somme indéfinie retournée, et les C_j sont des fractions rationnelles en n.

5. Retourner la récurrence

$$f(n) + C_1 f(n+1) + \cdots + C_J f(n+J) = G(n, n+1) - G(n, 0). \tag{4}$$

2.2. Version étendue de l'algorithme de Gosper. Les paramètres C_j ajoutés par Zeilberger interviennent de façon linéaire dans le polynôme $a(k)$ de la forme de Gosper-Petkovšek $a(k)/b(k)c(k+1)/c(k)$, et donc aussi dans l'équation de Gosper

$$a(k)x(k+1) - b(k-1)x(k) = c(k).$$

On applique donc le raisonnement usuel de l'algorithme de Gosper (cas 1, 2a et 2b) sauf qu'en plus des coefficients indéterminés de $x(k)$, on a aussi les coefficients $C_j(n)$.

2.3. Certificat. Outre la récurrence (4), l'algorithme de Zeilberger fournit un *certificat* $R(n,k) = G(n,k)/F(n,k)$, qui est une fraction rationnelle d'après le théorème 3, et qui permet de vérifier très facilement que cette récurrence est correcte. En effet, l'algorithme de Gosper étendu calcule $R(n,k)$ et des C_j tels que

$$\sum_k F(n,k) + C_1 F(n+1,k) + \cdots + C_J F(n+J,k) = G(n,k)$$

soit par définition de la sommation indéfinie par rapport à k,

$$\begin{aligned} F(n,k) + C_1 F(n+1,k) + \cdots + C_J F(n+J,k) \\ = R(n,k+1)F(n,k+1) - R(n,k)F(n,k). \end{aligned} \tag{5}$$

Pour vérifier une identité $\sum_{k=0}^{n} F(n,k) = C \neq 0$ à partir de son certificat $R(n,k)$, il suffit de former l'expression

$$F(n,k) - F(n+1,k) - R(n,k+1)F(n,k+1) + R(n,k)F(n,k),$$

de diviser tout par $F(n,k)$, ce qui donne une expression rationnelle dont on peut facilement tester la nullité.

3. L'algorithme de Berlekamp pour la factorisation dans $\mathbb{F}_p[x]$

Berlekamp fut le premier à proposer une méthode systématique pour factoriser des polynômes sur un corps fini **[2]** :

Algorithme de Berlekamp.
Entrée : un polynôme $f \in \mathbb{F}_p[x]$.
Sortie : l'ensemble des facteurs irréductibles de f et leur multiplicité.
1. [SQF] Effectuer une décomposition sans carré de f.
2. [DDF] Séparer les facteurs de degrés différents.
3. [EDF] Séparer les facteurs de même degré.

Factorisation sans carré. C'est l'étape la plus facile, car elle ne nécessite que des pgcds :

Théorème 4. *Soit $f(x)$ un polynôme unitaire de $R[x]$, où R est un domaine à factorisation unique de caractéristique zéro. Alors $f(x)$ est sans facteur carré si et seulement si* $\mathrm{pgcd}(f(x), f'(x)) = 1$.

```
>> f:=poly(x^11+x+1,IntMod(6449)): gcd(f,diff(f,x));

                    poly(1, [x], IntMod(6449))
```

On en déduit donc que $f = x^{11} + x + 1 \bmod 6449$ est sans facteur carré. Il existe d'autres méthodes plus efficaces pour calculer la décomposition sans carré, notamment l'algorithme de Yun [**3**]. Dans le cas de $R = \mathbb{F}_p$, une difficulé apparaît car $f'(x)$ peut être nul sans que $f(x)$ soit constant, par exemple $f(x) = x^{13} + 1$ dans $\mathbb{F}_{13}[x]$. Mais dans ce cas $f(x)$ est nécessairement de la forme $g(x)^p$.

Séparation des facteurs de degrés différents. Le théorème suivant peut servir à trouver le produit des facteurs de même degré.

Théorème 5. *Soit $f \in \mathbb{F}_p[x]$ sans facteur carré. Alors le produit des facteurs de f de degré divisant k est* $\mathrm{pgcd}(x^{p^k} - x, f) \bmod p$.

Si p^k est petit, on peut calculer directement ce pgcd :

```
>> d1:=gcd(poly(x^6449-x,IntMod(6449)),f);

                               2
          poly(2391 x + x  + 2155, [x], IntMod(6449))
```

Mais lorsque p^k devient grand, il vaut mieux calculer $x^{p^k} \bmod f$ par carrés et réductions successives modulo f avant d'en calculer le pgcd avec f.

```
>> f2:=divide(f,d1,Quo): xp2:=powermod(poly(x,IntMod(6449)),6449^2,f2):
>> d2:=gcd(xp2-poly(x,IntMod(6449)),f2);

                 2          3          4          5    6
poly(- 614 x - 167 x  - 2321 x  - 2101 x  - 2548 x  + x  - 395, [x], IntMod(6449))
```

Voici donc le produit des facteurs de degré 2 (il y en a trois). Dans cet exemple on a fini puisqu'il ne reste plus qu'un polynôme de degré trois, qui est forcément irréductible de par le théorème 5.

```
>> f3:=divide(f2,d2,Quo);

                               2    3
        poly(3211 x + 157 x  + x  + 150, [x], IntMod(6449))
```

Il nous reste donc à séparer les deux facteurs linéaires et les trois facteurs de degré deux.

Séparation des facteurs de même degré. Cantor et Zassenhaus [**4**] ont trouvé un algorithme probabiliste pour séparer les facteurs de même degré, basé sur l'identité $x^p - x = x(x^{(p-1)/2} - 1)(x^{(p-1)/2} + 1)$ pour p impair, qui impose aux facteurs de $x^p - x$ de se scinder entre $x^{(p-1)/2} - 1$ et $x^{(p-1)/2} + 1$.

Entrée : un polynôme f sans carré, produit de facteurs de degré i.
Sortie : les facteurs irréductibles de f.
0. Si $\deg f = i$, retourner f.
1. Générer un polynôme unitaire aléatoire $r \in \mathbb{F}_p[x]$ de degré $2i - 1$.
2. Calculer $q = r^{p^0} + r^{p^1} + \cdots + r^{p^{i-1}}$.
3. Calculer $g = \text{pgcd}(q^{(p-1)/2} - 1, f)$. Si $g \neq 1$ et $g \neq f$, recommencer en 1 avec g et f/g. Sinon recommencer avec f.

Théorème 6 **([4]).** *La probabilité que* $\text{pgcd}(q^{(p-1)/2} - 1, f)$ *soit non trivial est*

$$1 - \left(\frac{p-1}{2p}\right)^k - \left(\frac{p+1}{2p}\right)^k \geq \frac{4}{9}.$$

Donc on scinde le polynôme f à l'étape 3 avec une probabilité au moins $4/9$, ce qui donne une espérance finie pour le nombre total d'étapes.

```
>> z:=random(6449): r:=poly(z()+z()*x+z()*x^2+x^3,IntMod(6449));

                                   2     3
           poly(- 1711 x - 3055 x  + x  + 174, [x], IntMod(6449))

>> q:=r+powermod(r,6449,f2):
>> gcd(powermod(q,(6449-1)/2,f2)-q^0,f2);

                                   2
                poly(- 1372 x + x  + 2487, [x], IntMod(6449))
```

On a ainsi trouvé un facteur de degré 2, il suffit de recommencer avec le cofacteur de degré 4.

Le livre [**10**] donne une description plus détaillée. Il est à noter que l'étape de séparation des facteurs de degrés différents (DDF) a été notablement améliorée par Shoup [**17**]. Cette amélioration est implantée en MuPAD 1.3.

4. L'algorithme de Zassenhaus pour la factorisation dans $\mathbb{Z}[x]$

Factoriser un polynôme de $\mathbb{Q}[x]$ revient à factoriser un polynôme de $\mathbb{Z}[x]$, modulo la factorisation d'entiers. Nous supposons donc que le polynôme est à coefficient entiers. À titre d'exemple, nous considérons $f = x^8 + 4x^7 - 2x^6 - 20x^5 + 3x^4 + 44x^3 + 22x^2 - 4x + 34$ tiré de [**10**].

Factorisation sans carré. Comme pour l'algorithme de Berlekamp pour la factorisation dans $\mathbb{F}_p[x]$, on n'a besoin que de pgcds. De plus, comme on est en caractéristique zéro, on n'a pas de problème d'annulation de $f'(x)$.

Analyse des degrés possibles des facteurs. Contrairement à la factorisation dans $\mathbb{F}_p[x]$ (algorithme de Berlekamp), il n'est ici pas possible de séparer les facteurs de même degré. Par contre, on peut utiliser le fait que si $a(x)$ divise $f(x)$ dans $\mathbb{Z}[x]$, alors l'image de $a(x)$ divise l'image de $f(x)$ dans $\mathbb{F}_p[x]$ pour n'importe quel entier premier p. En particulier, si on trouve un p tel que $f(x)$ est irréductible dans $\mathbb{F}_p[x]$, alors $f(x)$ l'est aussi dans $\mathbb{Z}[x]$, comme $p = 2$ pour $x^9 + x + 1$.

Analyse des degrés possibles des facteurs.
Entrée : un polynôme $f \in \mathbb{Z}[x]$ de degré n et un entier premier p.
Sortie : les degrés possibles des facteurs de f.
1. Séparer les facteurs de degrés différents (DDF) de f dans $\mathbb{F}_p[x]$.
2. Si $d_1, \ldots, d_k$ sont les degrés obtenus, retourner $\bigcup d_{i_1} + \cdots + d_{i_j}$.

Remarque. On peut aussi imposer que f soit sans facteur carré dans $\mathbb{F}_p[x]$, ce qui revient à dire que $\mathrm{pgcd}(f, f') = 1$ dans $\mathbb{F}_p[x]$, ou encore que p ne divise pas $\mathrm{res}(f, f')$. En fait, on peut toujours trouver un tel p, car d'après l'inégalité d'Hadamard, le résultant de f et f' est borné par $n^{2n} 2^n B^{2n-1}$ où $n = \deg f$ et $B = \max |f_i|$.

Par exemple, pour $x^{10} + x + 1$, la factorisation modulo 2 donne $(x^3 + x + 1)(x^7 + x^5 + x^4 + x^3 + 1)$, donc comme degrés possibles $d_2 = \{3, 7\}$. Modulo 3, on trouve $x^{10}+x+1 \equiv (x-1)(x^3-x^2-x-1)(x^6-x^5+x^4-x^3+x+1)$, donc $d_3 = \{1, 3, 4, 6, 7, 9\}$, et l'intersection des degrés possibles est $d_{2,3} = d_2 \cap d_3 = \{3\}$; la factorisation modulo 5 donne $(x^2 - x + 2)(x^8 + x^7 - x^6 + 2x^5 - x^4 + 2x^2 + 2x - 2)$, à savoir $d_5 = \{2, 8\}$, donc $d_{2,3,5} = \varnothing$, dont on déduit que $x^{10} + x + 1$ est irréductible.

Même lorsque f n'est pas irréductible, les informations sur les degrés possibles seront utiles par la suite. Pour notre polynôme f, la factorisation modulo 5, à savoir $f = (x^2 - x + 1)(x^2 - 2x - 2)(x^2 - x + 2)(x^2 - 2x - 1)$, montre que les facteurs possibles ont degré dans $d_5 = \{2, 4, 6\}$.

Remontée de Hensel. La remontée de Hensel (*Hensel-lifting*) consiste à « remonter » une factorisation dans $\mathbb{F}_p[x]$ en factorisation dans $\mathbb{F}_{p^N}[x]$. C'est un peu comme le théorème des restes chinois, sauf qu'ici on a un seul nombre premier.

Remontée de Hensel
Entrée : une factorisation $f = gh$ modulo p^k avec g, h premiers entre eux et h primitif.
Sortie : une factorisation $f = g^* h^*$ modulo p^{k+1} avec $g^* = g \bmod p^k$ et $h^* = h \bmod p^k$.
0. Calculer s et t tels que $sg + th = 1$ par l'algorithme d'Euclide étendu.
1. Calculer $e = f - gh \bmod p^{k+1}$.
2. Effectuer la division euclidienne de se par h : $se = qh + r \bmod p^{k+1}$

avec $\deg r < \deg h$.

3. Retourner $g^* = g + te + qg \bmod p^{k+1}$ et $h^* = h + r \bmod p^{k+1}$.

Supposons que l'on prenne $h = x^2 - x + 1$ et $g = (x^2 - 2x - 2)(x^2 - x + 2)(x^2 - 2x - 1) = x^6 + 2x^4 + 2x^3 - 2x^2 - 1 \bmod 5$. La remontée de 5^1 vers 5^2 donne $g^* = x^6 + 12x^4 + 7x^3 - 7x^2 - 5x - 6$ et $h^* = x^2 + 4x + 11$.

***Théorème 7* (Unicité de la remontée de Hensel).** *Soit p un entier premier, N un entier positif, $f, g_1, h_1, g_2, h_2 \in \mathbb{Z}[x]$ avec h_1, h_2 primitifs et de même degré, $g_1 \equiv g_2 \bmod p$, $h_1 \equiv h_2 \bmod p$, $\mathrm{pgcd}(g_1, h_1) = 1 \bmod p$, et $f \equiv g_1 h_1 \equiv g_2 h_2 \bmod p^N$. Alors $g_1 \equiv g_2 \bmod p^N$ et $h_1 \equiv h_2 \bmod p^N$.*

Algorithme de Zassenhaus [18].
Entrée : un polynôme $f \in \mathbb{Z}[x]$ primitif et sans carré de degré n.
Sortie : deux polynômes $g, h \in \mathbb{Z}[x]$ primitifs et non constants tels que $f = gh$ ou bien « f est irréductible ».
1. Choisir un entier premier p ne divisant pas $\mathrm{res}(f, f')$, et $N = \lceil \log_p(2^n\,|f|) \rceil$.
2. Calculer une factorisation $f = w_1 \cdots w_k$ de f dans $\mathbb{F}_p[x]$.
   Si $k = 1$, retourner « f est irréductible ».
3. Effectuer les étapes 4 et 5 pour chaque sous-ensemble S de $\{1, \ldots, k\}$ ayant entre 1 et $k/2$ éléments.
4. Soit $g_1 = \prod_{i \in S} w_i$ et $h_1 = \prod_{i \notin S} w_i$, calculer s, t tels que $sg_1 + th_1 = 1 \bmod p$.
5. Calculer $g_N, h_N \in \mathbb{Z}[x]$ tels que $g_N \equiv g_1 \bmod p$, $h_N \equiv h_1 \bmod p$ et $f \equiv g_N h_N \bmod p^N$.
   Si $f = g_N h_N$ retourner g_N, h_N.
6. Retourner « f est irréductible ».

À l'étape 4, si le degré de g_N ne fait pas partie des degrés possibles issus de l'analyse préliminaire, alors on peut passer directement au sous-ensemble suivant. La borne de Mignotte donnée par le théorème suivant garantit que la valeur de N choisie à l'étape 1 est assez grande pour garantir l'exactitude du résultat « f est irréductible » en 6.

***Théorème 8* (Mignotte 1989).** *Soit $g = \sum g_i x^i \in \mathbb{C}[x]$ un facteur de degré m de $f = \sum f_i x^i \in \mathbb{C}[x]$ de degré n. Alors*

$$\|g\| \leq \left|\frac{g_m}{f_n}\right| 2^n \|f\|$$

où $\|f\|$ représente la norme euclidienne de f.

L'algorithme de Zassenhaus est de complexité exponentielle dans le cas le pire, par exemple avec les polynômes de Swinnerton-Dyer

$$f_k = \prod (x \pm \sqrt{2} \pm \sqrt{3} \pm \cdots \pm \sqrt{p_k})$$

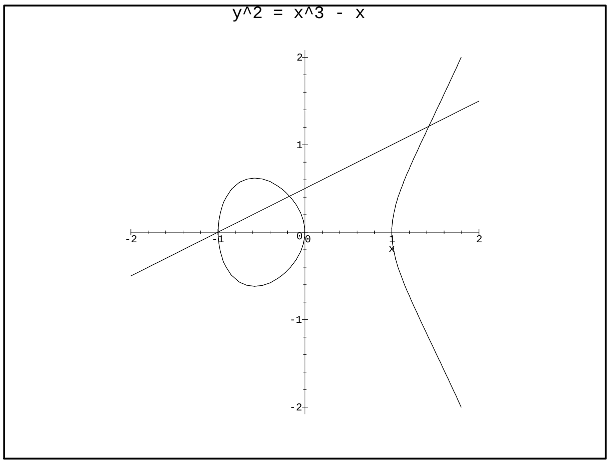

FIGURE 1. La courbe elliptique $y^2 = x^3 - x$

où p_k est le ke nombre premier. Le polynôme f_k est à coefficients entiers, irréductible sur $\mathbb{Z}$, mais se décompose en facteurs de degré au plus deux sur $\mathbb{F}_p$ pour tout nombre premier p ! Cependant Collins a montré que l'algorithme de Zassenhaus se comporte « en moyenne » de façon polynomiale ; c'est encore aujourd'hui l'algorithme utilisé en pratique par les systèmes de calcul formel.

5. L'algorithme ECM de Lenstra pour la factorisation d'entiers

Définition 4. Soit F un corps de caractéristique différente de 2 et 3 et $x^3 + ax + b$ un polynôme de $F[x]$ sans racines multiples. Alors $E = \{(u, v) \in F^2, v^2 = u^3 + av + b\} \cup \{\mathcal{O}\}$ est une courbe elliptique sur F, où $\mathcal{O}$ est le « point à l'infini » sur E.

Exemple. Le polynôme $p = x^3 - x$ n'a pas de racines multiples si $\operatorname{car} F \neq 2$, car $\operatorname{res}(p, p') = -4$. Donc $y^2 = x^3 - x$ définit une courbe elliptique sur $\mathbb{R}$ ou $\mathbb{C}$.

Le polynôme $p = x^3 + ax + b$ n'a que des racines simples si le résultant de p et de p' (i.e. son discriminant) est non nul, ce qui revient à dire que $4a^3 + 27b^2$ ne s'annule pas.

```
>> p:=x^3+a*x+b: resultant(p,diff(p,x),x);

                         3         2
                      4 a   + 27 b
```

L'équation $y^2 = x^3 + ax + b$ est l'équation de Weierstrass de la courbe E, et a, b sont ses coefficients de Weierstrass.

Le point à l'infini se comprend mieux sur l'équation projective, où on ajoute une troisième coordonnée $z = 1$ pour les points « standards » :

$$y^2 z = x^3 + axz^2 + bz^3$$

et $x = 0, y = 1, z = 0$ pour le point à l'infini $\mathcal{O}$.

Addition sur E. On définit une structure de groupe sur une courbe elliptique de la façon suivante :

Définition 5. Soient P_1 et P_2 deux points sur une courbe elliptique E, et R le troisième point d'intersection de la droite (P_1, P_2) avec E. Alors si $R = (x, y)$, on définit $P_1 + P_2 = (x, -y)$. Dans le cas particulier où $P_1 = P_2$, on considère la tangente en P_1. Si $P_1 = \mathcal{O}$, alors on prend la droite verticale passant par P_2.

Dans le cas $P_1 = \mathcal{O}$, R est le symétrique de P_2 par rapport à l'axe horizontal, donc $\mathcal{O} + P_2 = P_2$: $\mathcal{O}$ est l'élément neutre du groupe. L'opposé de $P = (x, y)$ pour l'addition dans E est donc $(x, -y)$.

Les coodonnées (x_3, y_3) de $P_1 + P_2$ en fonction de celles de P_1 et de P_2 sont :

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \qquad y_3 = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3).$$

Exemple. Sur F_5, la courbe $y^2 = x^3 - 2$ a exactement six points $a = \mathcal{O}$, $b = (3, 0)$, $c = (2, 1)$, $d = (1, 2)$, $e = (1, 3)$, $f = (2, 4)$.

***Théorème 9* (Hasse)**. *Le nombre n de points d'une courbe elliptique sur $\mathbb{F}_q$ vérifie*

$$|n - (q + 1)| \leq 2\sqrt{q}.$$

Dans l'algorithme de Lenstra, on utilise des pseudo courbes elliptiques dans $\mathbb{Z}/N\mathbb{Z}$ (qui n'est pas un corps).

> **Algorithme de Lenstra. [13]**
> Entrée : un entier composite N non divisible par 2 et 3, ni une puissante parfaite.
> Sortie : un facteur non trivial de N ou « échec ».
> 1. Choisir des entiers positifs B (borne sur les nombres premiers de la base) et C (borne sur un facteur de N). Soit
>
> $$k = \prod_{\substack{r \text{ premier} \\ r \leq B}} r^{\alpha_r}$$
>
> où $\alpha_r \in \mathbb{N}$ est déterminé par $r^{\alpha_r} \leq C + 2\sqrt{C} + 1 < r^{\alpha_r + 1}$.
> 2. Choisir au hasard une « courbe elliptique » E sur $\mathbb{Z}_N$ et un point P sur E, et calculer kP, en utilisant les formules d'addition sur la « courbe elliptique » :
>
> $$kP = r \cdots r \cdots 3 \cdots 3 \cdot 2 \cdots 2 \cdot P$$

et en calculant de droite à gauche. Pour cela, on a besoin de faire des divisions $u/v \bmod N$; si $g = \mathrm{pgcd}(v, N) = 1$, alors on peut inverser v et continuer, sinon on a trouvé un diviseur non trivial de N que l'on retourne.

3. Si on arrive à calculer $kP = \mathcal{O}_E$, alors la factorisation est sans succès.

En notant $L(x) = e^{\sqrt{\log x \log\log x}}$, la valeur optimale de B est $B \simeq L(p)$ où p est le plus petit facteur premier de N, et le temps de calcul de l'algorithme de Lenstra est alors de l'ordre de $L(p)^2$.

L'algorithme ECM de Lenstra n'est pas implanté dans MuPAD 1.3, où par contre est implanté l'algorithme ECPP — utilisant aussi les courbes elliptiques — de preuve de primalité dû à Atkin et Morain [**1**] (fonction `numlib::proveprime`).

Références

[1] Atkin, A. O. L., & Morain, F. Elliptic curves and primality proving. *Mathematics of Computation 61*, 203 (July 1993), 29–68.

[2] Berlekamp, E. R. Factoring polynomials over finite fields. *Bell System technical Journal 46* (1967), 1853.

[3] Bronstein, M. *Symbolic Integration I. Transcendental Functions*, vol. 1 of *Algorithms and Computation in Mathematics*. Springer, 1997.

[4] Cantor, D. G., & Zassenhaus, H. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation 36* (1981), 587–592.

[5] Cornil, J.-M., & Testud, P. *Maple, introduction raisonnée à l'usage de l'étudiant, de l'ingénieur et du chercheur*. Springer-Verlag, 1995. 463 pages.

[6] Davenport, J. H., & Singer, M. F. Elementary and Liouvillian solutions of linear differential equations. *Journal of Symbolic Computation 2* (1986), 237–260.

[7] Davenport, J. H., Siret, Y., & Tournier, E. *Calcul formel*, 2nd ed. Masson, Paris, 1993.

[8] Flajolet, P., & Odlyzko, A. M. Singularity Analysis of Generating Functions. *SIAM Journal on Discrete Mathematics 3*, 2 (1990), 216–240.

[9] Flajolet, P., Salvy, B., & Zimmermann, P. Automatic Average–case Analysis of Algorithms. *Theoretical Comput. Sci. 79*, 1 (Feb. 1991), 37–109.

[10] Geddes, K. O., Czapor, S. R., & Labahn, G. *Algorithms for computer algebra.* Kluwer Academic Publishers, 1992.

[11] Gomez, C., Salvy, B., & Zimmermann, P. *Calcul formel : mode d'emploi. Exemples en Maple.* Masson, 1995.

[12] Gosper, R. W. Decision procedure for indefinite hypergeometric summation. *Proceedings of the National Academy of Sciences USA 75*, 1 (Jan. 1978), 40–42.

[13] Lenstra, H. W. Factoring integers with elliptic curves. *Annals of Mathematics 126* (1987), 649–673.

[14] Leroux, A., & Pomès, R. *Applications de Maple.* Vuibert, 1995. ISBN-2-7117-8859-8.

[15] Petkovšek, M., Wilf, H., & Zeilberger, D. *A=B.* A. K. Peters, Ltd., Wellesley, Mass., 1996.

[16] SALVY, B. *Asymptotique automatique et fonctions génératrices*. Thèse de doctorat, École Polytechnique, 1991.

[17] SHOUP, V. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation 20*, 4 (Oct. 1995), 363–397.

[18] ZASSENHAUS, H. On Hensel factorization I. *Journal of Number Theory 1* (1969), 291–311.

[19] ZEILBERGER, D. The method of creative telescoping. *Journal of Symbolic Computation 11* (1991), 195–204.

PAUL ZIMMERMANN, INRIA Lorraine • *E-mail :* `Paul.Zimmermann@inria.fr`