Algorithmes de factorisation de polynômes

Denis Lugiez

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE
«Informatique»

par

Denis LUGIEZ

ALGORITHMES DE FACTORISATION DE POLYNOMES.

Thèse soutenue le 25 janvier 1984 devant la commission d'examen.

G. VEILLON	Président
J. CALMET	Examinateurs
J.C. LATOMBE	
D. LAZARD	
R. LOOS	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Je tiens à remercier ,

Monsieur G. VEILLON , professeur à l'Institut National Polytecthnique de Grenoble , d'avoir bien voulu présider ce jury.

Monsieur J.C. LATOMBE , professeur à l'I.N.P.G. d'avoir accepté d'examiner et de juger ce travail.

Monsieur D. LAZARD , professeur à l'Université de PARIS VI, d'avoir accepté de juger ce travail et qui m'a fait bénéficier de sa grande expérience du sujet.

Monsieur R. LOOS , professeur à l'Université de KARLSRUHE , d'avoir accepté de venir juger ce travail .

Tout particulièrement Jacques Calmet qui a su m'accueillir avec chaleur au sein de l'équipe, petite mais symphatique , de CALCUL FORMEL du L.I.F.I.A. et qui a su me proposer un sujet adapté à ma formation primitive de mathématicien . Je le remercie ,en particulier pour les discussions constructives que j'ai eu avec lui, et pour les possibilités d'échange que j'ai pu avoir au cours de rencontres scientifiques.

Monsieur P. JORRAND, responsable de la formation INFORMATIQUE de l'I.N.P.G. et directeur du L.I.F.I.A., pour mon accueil dans ce laboratoire et cette spécialité.

Andrée CHAPEL qui a permis , en récupérant des disquettes endommagées , que cette thèse soit prete à temps.

Monsieur D. IGLESIAS et le service de reprographie de l'IMAG pour le tirage de cette thèse.

- INDEX -

INTRODUCTION

Ce travail a pour objet l'etude d'algorithmes de factorisation de polynomes à coefficients entiers et traite plus spécialement les methodes p-adiques utilisant le lemme de Hensel. Le lien avec la decomposition d'une fraction rationnelle en éléments simples nous permet de proposer de nouveaux algorithmes et les comparaisons avec les methodes classiques sont effectuees. La factorisation des polynomes est un des points importants des systemes de calcul formel ,et avant d'entrer dans le vif du sujet il convient de preciser la notion un peu vague de calcul formel.

1-LE CALCUL FORMEL

Le nom est une traduction très libre de l'anglais "computer algebra",mais à ses débuts la discipline était baptisée " Manipulations Algèbriques et Symboliques".Il s'agit en fait d'une discipline informatique récente où les mathematiques,surtout l'algèbre jouent un grand role. Cela l'a longtemps empeche d'etre reconnue comme une discipline à part entière, alors que ses possibilités sont vastes et son avenir prometteur.

Le but du calcul formel est d'une part l'étude et la réalisation d' algorithmes effectuant des calculs algèbriques, d'autre part la manipulation d'objets dits symboliques. Le domaine de ses travaux est vaste car en plus des problemes d'algorithmiques algebriques, il comprend la reconnaissance des formes, la simplification, la demonstration automatique,...,ce qui justifie la place du calcul formel dans le domaine de l'intelligence artificielle. Il faut reconnaitre que la discipline s'est d'abord préoccupée des algorithmes algébriques, sans pour autant négliger l'aspect informatique fondamentale, mais à présent la nouvelle génération de logiciels en cours de realisation , plus axée sur la représentation des connaissances mathématiques et leur utilisation, va relancer l'aspect informatique de la discipline.

Néanmoins les algorithmes algébriques resteront la base de tels systèmes et un des champs de recherche les plus riches du calcul formel. Une bonne synthèse des connaissances actuelles en cette matière peut se trouver dans "Computer Algebra" edite par Buchberger,Collins,Loos (B.C.L. 82) . De nombreux problèmes proviennent de la physique et les physiciens sont de grands utilisateurs du calcul formel, certains n'ont d'ailleurs pas hesite à écrire leur propre système.(Wolfram avec S.M.P.)

L'aspect informatique des systèmes va et doit évoluer. En effet le langage et la structure utilisée se répercutent au niveau de la facilite d'emploi, de la transparence et des performances (rapidite,place memoire,...) et de gros progrès sont à attendre en ce qui concerne la modularité et la transparence. Or les progrès technologiques et théoriques vont bouleverser la conception des nouveaux systèmes de calcul formel. On peut parler d'une nouvelle génération de logiciels alliant les derniers progrès en intelligence artificielle à ceux accomplis en algorithmique , mis en oeuvre par un langage encore à ecrire. De tels systèmes seront à la portée du plus grand nombre et , grace aux progrès techniques ils seront disponibles sur de petits ordinateurs tout en restant performants. Cependant l'aspect algorithmique reste essentiel, surtout si on désire un système aussi complet que possible. C'est donc cet aspect qui est étudié ici, et il n'est pas inutile de s'apercevoir que de nombreux problèmes restent posés et que le choix de bons algorithmes n'est pas évident à priori.

2-LA FACTORISATION DES POLYNOMES

Il s'agit de polynomes à coefficients entiers qu'on factorise en un produit de polynomes irréductibles sur Z. Le problème est différent de celui de la factorisation sur les réels ou les complexes pour lesquels il existent des méthodes éprouvées basée sur la recherche des racines. Le problème n'est pas nouveau et tout le monde a factorisé des polynomes simples à l'aide des identités remarquables bien connues $(x - a)(x + a)$. Dès qu'on aborde des cas moins triviaux le problème se complique mais il est résolu depuis longtemps.

Newton dès 1707, donne une méthode qui permet de calculer les facteurs lineaires et quadratiques ,puis en 1793 un astronome Pit Schubert étend sa méthode qui permet donc de factoriser un polynome en un nombre fini d'opérations. Kronecker redécouvrait la méthode 90 ans plus tard et l'exposait clairement. Mais dès que le degré du polynome ou la taille des coefficients augmente cette méthode s'avère totalement inefficace. Le progrès décisif viendra de l'algorithme découvert par Berlekamp pour factoriser un polynome à coefficients dans un corps fini Z_p et de l'emploi de méthodes p-adiques.

L'idée décisive est de factoriser le polynome dans un corps fini $Z_p[x]$, puis de remonter cette factorisation dans un anneau $Z_{p^k}[x]$ jusqu'à ce que p soit assez grand. Ensuite il n'y a plus qu'à recombiner les facteurs obtenus entre eux pour obtenir les facteurs réels du polynome. La remontée des facteurs s'effectue à

l'aide du lemme de Hensel exposé en 1900, une autre possibilite est d'utiliser le théorème des restes chinois. D'autre méthodes basées sur des algorithmes probabilistes et l'utilisation de grands nombres pseudo-premiers sont apparues récemment, mais elles ne sont qu'à leur début.

Un des points les plus interessants des méthodes actuelles est l'utilisation des méthodes p-adiques qui permettent de passer d' une solution partielle à une vraie solution. Suivant une vieille tradition on résoud un problème compliqué en résolvant un problème plus simple puis à l'aide de cette solution partielle on calcule la solution du problème initial. Les méthodes p-adiques sont utilisées couramment dans d'autre cas : calcul de P.G.C.D., décomposition de fractions rationnelles...

3-LES METHODES P-ADIQUES

La base de ces méthodes est le lemme de Hensel qui permet de remonter une égalité $c = a.b$ modulo p en une égalité $c' = a'.b'$ modulo p , où c,a,b sont les images de c',a',b' modulo le nombre premier p ; à condition que a et b vérifient une identité de Bezout du type $a.u + b.v = 1 \bmod p$ avec $\deg(v) < \deg(a)$.

A partir de cette idée de nombreuses variantes sont possibles:

- Remontée quadratique : on remonte les égalités en élevant à chaque fois le module au carré, le module final est du type p .

- Remontée lineaire : on remonte suivant les puissances successives de p.

- Remontée parallèle : on remonte une égalité $c = a_1 \ldots a_n$ en remontant tous les facteurs a_i simultanément.

L'interet de ces méthodes est de rejeter les calculs pénibles dans $Z[x]$ en des calculs plus faciles dans le corps $Z_p[x]$, p étant un petit nombre premier. Par exemple l'explosion de la taille des coefficients qui se produit lors des calculs de P.G.C.D. dans $Z[x]$ est évitée. De plus le fait de travàiller modulo une puissance de p lors de la remontée permet de controller la taille des coefficients. Ainsi un des problèmes majeur du calcul sur les polynomes à coefficients entiers est surmonté. Bien entendu il faut pouvoir passer des résultats modulo p au résultats sur les entiers. A ce niveau interviennent les

bornes calculées à l'avance. Il est possible de borner la valeur absolue des coefficients des facteurs d'un polynome donné, par suite dès que ces bornes sont dépassées on s'arrete. Ces résultats peuvent etre généralisés à des anneaux plus généraux que Z, voir Lauer (LA 83) .

Les méthodes p-adiques présentent certains inconvénients dus au fait que la structure de Zp[x] est plus riche que celle de Z[x] donc un polynome irréductible de Z[x]peut devenir réductible dans Zp[x]. Mais ces inconvénients sont largement compensés par les avantages procurés par les méthodes p-adiques.

4-BUT ET PLAN DE CE TRAVAIL

Il ne s'agit pas ici de donner une liste exhaustive des algorithmes de factorisation de polynomes, mais plutot d'illustrer les tendances actuelles de recherche dans le domaine de la factorisation. Notament on s'est efforcé de montrer les différentes approches retenues pour résoudre les problèmes particuliers de la factorisation (coefficient de tete, facteur parasite,...).

On s'est attaché à comparer les différents algorithmes de remontée pour déterminer les plus efficaces en pratique, surtout quand il s'agit des nouveaux algorithmes que nous proposons ici. Car les concepteurs de système ne retiennent pas les algorithmes qui sont les meilleurs en théorie mais ceux qui sont en pratique les plus efficaces. Cependant nous ne négligeront pas les études de complexité qui permettent une estimation théorique du cout des algorithmes.

Les algoritmes sont décrits dans un langage symbolique dont le but est de représenter de manière compréhensible les algorithmes et dont la sémantique est semblable à celle d'ALGOL.

Le chapitre 1 présente deux algorithmes de factorisation sur $Z_p[x]$.Le premier est l'algorithme classique de Berlekamp qui est sans doute le plus utilisé. Certains éléments de sa démonstration sont donnés pour illustrer les type de raisonnements utilisés. Le second est celui de Camion-Lazard qui est un algorithme probabiliste. Cela permet d'une part d'expliquer cette notion et d'indiquer certains points de recherche actuels.

La factorisation sur les entiers est l'objet du chapitre 2. les procédés habituels de simplification du problème sont donnés

ainsi que la méthode classique de Berlekamp-Hensel. Les problèmes généraux de la factorisation sont évoqués en particulier celui des facteurs parasites, et l'algorithme récent de Lenstra qui résoud ce problème est décrit. D'autre méthodes de factorisation utilisant les restes chinois ou la factorisation modulo un grand nombre premier sont abordées. Les résultats de Calmet-Loos sur la comparaison de ces méthodes sont rappelés.

Le chapitre 3 aborde les méthodes p-adiques et traite des différentes stratégies possibles pour appliquer le lemme de Hensel, parallèle ou série, linéaire ou quadratique. Le lien avec les fractions rationnelles est décrit et un nouvel algorithme basé sur cet aspect est donné. Les résultats de comparaisons numériques permettent d'estimer les possibilités de chaque algorithme.

Enfin la factorisation dans le cas de plusieurs variables fait l'objet du chapitre 4. Les problèmes spécifiques à ce cas sont décrits ainsi qu'une première forme de remontée. La forme améliorée de cet algorithme est ensuite proposée ce qui permet de voir les méthodes utilisées pour résoudre ces problèmes. Un nouvel algorithme utilisant les fractions rationnelles, comme Viry le suggérait, les avantages qu'on peut en attendre sont exposés. Des comparaisons numériques suivent ainsi que des remarques sur les problèmes soulevés par la programmation.

Les perspectives sur le lemme de Hensel généralisé sont rapidement évoquées, ainsi que l'évolution possible de la résolution du problème de la factorisation. L'aspect de programmation est abordé en particulier avec la liste des programmes écrits pour les nouveaux algorithmes.

CHAPITRE 1

FACTORISATION DES POLYNOMES A COEFFICIENTS ENTIERS DANS Zp[X] p ENTIER PREMIER

1- PRESENTATION ET POSITION DU PROBLEME

Ce chapitre est consacré aux méthodes de factorisation dans Zp[x], où Zp est le corps fini à p éléments, p étant un nombre premier. Nous ne traitons pas le cas général de la factorisation dans Zq[x], avec q une puissance de p. En particulier les algorithmes de Rabin (Ra 80) pour la recherche de racines dans de tels corps ne seront pas décrits. Mais il faut savoir que la plupart des résultats donnés dans Zp[x] sont valables dans Zq[x].

Le but de ce chapitre n'est pas d'etre exhaustif, mais de donner la méthode la plus classique, celle de Berlekamp et d'indiquer les lignes actuelles de recherche en donnant l'algorithme de Camion-Lazard (Ca 82,La 82), qui dérive d'algorithmes décrit par Cantor et Zassenhauss (CA ZA 81).

Les polynomes à factoriser dans Zp[x] sont des polynomes sans carrés, i.e. sans facteurs multiples. Les méthodes pour se ramener à de tels polynomes sont exposées dans le chapitre 2 . On supposera que le polynome de Z[x] dont il est l'image par l'homomorphisme naturel est primitif, i.e. que le P.G.C.D. de ses coefficients est 1 ou -1. De plus dans le corps Zp, le coefficient de tete du polynome est inversible, donc nous supposerons que le polynome est unitaire, et par suite ses facteurs seront aussi unitaires.

2 - FACTORISATION EN PRODUIT DE POLYNOMES IRREDUCTIBLES

2.1 Définition

Soit P le polynome à factoriser, il s'écrit $P = \prod_{i=1,..,n} Qi$ avec n le degré de P et Qi = 1 ou bien Qi est le produit des facteurs irréductibles de P de degré i.

Cette factorisation s'appelle la factorisation de P en produit d'irréductibles de meme degré.

Elle permet de transformer la factorisation de P en plusieurs factorisations plus simples dans lesquelles tous les facteurs irréductibles ont le meme degré. Elle permet de conclure si le polynome est irreductible et parfois meme d'obtenir une factorisation complète.

Exemple : $P(x) = (x + 1)(x + 7)(x^5 + x^3 + x^2 + x + 1) \bmod 13$

On trouve deux facteurs non triviaux:

$$x^2 + 8x + 7 \quad \text{et} \quad x^5 + x^3 + x^2 + x + 1$$

$x^2 + 8x + 7$ se factorise en calculant les racines.

$x^5 + x^3 + x^2 + x + 1$ s'avère irréductible car il ne peut etre produit de facteurs de meme degré i pour i = 1 , 2 , 3 , 4 donc il est produit de facteurs linéaires et on montre qu'il n'a pas de racines dans Zp pour p = 13.

Le polynome P est irréductible si tous les facteurs Qi obtenus valent 1 sauf Qn égal à P.

2.2 Théorème fondamental

Il s'agit du théorème qui est à la base de l'algorithme et qui permet d'obtenir d'autres résultats interessants.

Théorème: Dans Zp[x], le polynome $x^{p^m} - x$ est égal au produit de tous les polynomes irréductibles unitaires dont le degré divise m.

démonstration: On démontre d'abord que si f est un polynome irréductible de degré d alors f divise $x^{p^{kd}} - x$, avec k un entier positif.

f étant irréductible , $K = Zp[x]_d/(f)$ l'ensemble des classes de résidus modulo f est un corps à p^d éléments.

Donc tout élément z de ce corps vérifie l'équation fondamentale :

$$x^{p^d} - x = 0$$

Plus généralement z vérifie : $x^{p^{kd}} - x = 0$

Prenons pour z la classe de résidu particulière qui contient le polynome x, alors f(z) = 0 mod f car f(z(x)) = f(x).

Dans K , z est racine du polynome f et il est aussi racine du polynome $h(x) = x^{p^{kd}} - x$.

La division de h par f s'écrit: $h(x) = f(x).q(x) + r(x)$

avec $deg(r) < deg(f)$

Pour z égal à x on a r =0 , comme r est un polynome de degré inferieur à celui de f , alors r = 0.

Donc on a le résultat cherché, f divise $x^{p^{kd}} - x$.

Montrons maintenant le résultat définitif :

Le produit des polynomes irréductibles dont le degré divise m est égal à : $x^{p^m} - x$.

D'après ce qui précède , tout polynome irréductible unitaire dont le degré divise m ,divise $x^{p^m} - x$.

Soit Ik le nombre de polynomes irréductibles unitaires de degré k. Alors le degré du produit des polynomes irréductibles unitaires dont le degré divise m est $\sum_{k/m} k.Ik$

On sait que $\sum_{k/m} k.Ik = p^m$ voir Berlekamp (Be 68) p79

La conclusion vient sans efforts :

$x^{p^m} - x$ est un polynome unitaire de degré p^m qui est divisible par un polynome unitaire de degré p^m, donc ces deux polynomes sont égaux.

C.Q.F.D.

2.3 Algorithme de factorisation partielle

Une fois que le théorème cité en 2.2 est établi l'algorithme de factorisation partielle en découle naurellement. L'idée de base est simple: puisque $x^{p^m} - x$ est le produit des polynomes irréductibles unitaires de degré i, il suffit de calculer le P.G.C.D. de $x^{p^m} - x$ et du polynome à factoriser, pour obtenir le produit des facteurs irréductibles de degré i de P.

Algorithme de factorisation partielle.

Entrée : p un nombre premier , P le polynome unitaire sans carré, primitif à factoriser , n son degré.

Sortie : L liste des couples (Qi,i) où Qi est le produit des facteurs unitaires irréductibles de P de degré i.

(1) L = ∅

(2) Tant que i $\leqslant \lfloor n/2 \rfloor$ Faire

début

(21) $Qi = PGCD(x^{p^m} - x, P)$

(22) Si Qi = 1 alors L = L∪(Qi,i) , P ← P/Qi

(23) i = i + 1

fin

(3) Si deg(P) > 0 alors L = L∪ (P, deg(P))

fin.

Preuve : Elle découle directement du théorème précédent. Si Hi est le produit des facteurs irréductibles de P de degré i alors $Hi = PGCD\ (x^{p^m} - x, P)$.

Mais Hi est premier avec tout facteur irréductible de P de degré j i , donc le PGCD ne change pas si on remplace P par le quotient de P par ses facteurs irréductibles de degré j. Donc tous les Hi sont trouvés par l'algorithme pour j de 1 à n/2. Si P a un facteur irréductible de degré supérieur H , celui-ci a un cofacteur de degré inferieur , donc H = P/ ΠHi ; c'est ce qui est calculé à l'étape 6.

C.Q.F.D.

Remarques : - L'obtention des facteurs irréductibles de degré supérieur à n/2 illustre une technique classique de la factorisation. On calcule les facteurs de degré inférieur à n/2 et les autres facteurs sont obtenus comme cofacteurs d'un facteur déjà trouvé.

- Le degré de x^{p^m} est très élevé donc le calcul du PGCD de $x^{p^m} - x$ et de P coute cher. Aussi on remplace x^{p^m} par x^{p^m} modulo P

Le résultat est inchangé car :

$G = PGCD(x^{p^m} - x, P) \Leftrightarrow A.(x^{p^m} - x) + B.P = G$ avec $deg(A) < deg(P)$

$\Leftrightarrow A.((Q.P + R) - x) + B.P = G$

si Q.P + R est la division de $x^{p^m} - x$ par P.

$\Leftrightarrow A.(R - x) + B'.P = G$

$\Leftrightarrow$ G PGCD de R-x et de P

L'algorithme effectivement implanté dans les systèmes est :

Entrée : P polynome primitif sans carré unitaire de degré n

Sortie : L liste des couples (Qi,i) avec Qi produit des facteurs irrédutibles de degré i ou Qi = 1 .

(1) L = ∅ , i = 1

(2) Tant que i < n/2 Faire
début
(21) $h = x^{p^m} \bmod P$
(22) Qi = PGCD(h-x,P)
(23) Si Qi = 1 alors L = Lu(Qi,i) , P ← P/Qi
(24) i = i+1
fin
(3) Si deg(P) > 0 alors L = Lu(P,deg(P))
fin

2.4 COMPLEXITE

2.4.1 Définition

Nous rappelons certaines définitions:

- M(n) est le temps requis pour la multiplication de deux polynomes de degré n.

$M(n) = n^2 L^2(p)$ si p majore la valeur absolue des coefficients des polynomes. $L^2(p)$ représente le logarithme de p dans une base

convenable , en général égale au mot machine. Cette expression de M(n) est celle obtenue avec les méthodes classiques de multiplication.

$M(n) = n \text{ Log}(n) L^2(p)$ pour les méthodes utilisant la transformée de Fourier rapide (FFT).

- D(n) est le temps requis pour la division de deux polynomes de degré n. Il vaut O (M(n)) d'après un résultat classique qu'on peut trouver dans Aho , Hopcroft et Ullman (AH HO UL).

- GCD(n) est le temps de calcul du PGCD de deux polynomes de degré n ; il vaut $O(n^2 L^2(p))$ dans Zp[x].

2.4.2 Cout de l'algorithme de factorisation partielle.

Les étapes à considerer sont les suivantes :

(4) Calcul de Qi : Cout GCD(n) effectué n/2 fois.

(5) Réduction de P : Cout D(n) effectué n/2 fois au plus.

(3) Calcul de x^{p^i} modulo P

Cette dernière étape nécessite une approche plus fine. Une manière interessante d'effectuer ce calcul est d'utiliser la matrice Q de Berlekamp.

(6) Matrice Q de Berlekamp : C'est la matrice (n,n) dont la i° ligne vaut x^{p^i} modulo P.

Si on représente un polynome par un vecteur alors pour $H = x^p$ on a :

$$Q.H = (H)^p$$

On utilise l'identité de Fermat pour les polynomes pour montrer ce résultat , soit :

$$g(x^p) = (g(x))^p.$$

Donc les puissances successives de x sont calculées par multiplications successives par la matrice Q , ce qui donne à chaque fois n^2 multiplications dans Zp.

Le cout de l'étape (3) est donc : $O (n^3 L^2(p))$.

(b) Cout de construction de la matrice Q :

- Calcul de x^{p^i} modulo P ; on utilise un arbre, à l'étape i on calcule x^i comme $(x^{(i/2)})^2$.

Par exemple pour p = 8, on effectue les calculs suivants:

calcul de x^8 mod P:

calcul de x^2 mod P par carré de x mod P

calcul de x^4 mod P par carré de x^2 mod P

calcul de x^8 mod P par carré de x^4 mod P

Le calcul de x^p modulo P s'effectue en multipliant $x^{p(i-1)}$ mod P par x^p mod P.

Le cout total de construction de la matrice Q est donc de

$O (n^2 L^2(p)Logp + n^3L^2(p))$ pour les méthodes classiques.

on peut alors énoncer la proposition suivante :

Proposition . Le cout de l'algorithme de factorisation partielle est :

$$O(n^2L^2(p) + n^3 L^2(p))$$

pour les méthodes classiques.

3 - METHODE DE BERLEKAMP.

La méthode de Berlekamp qui permet de factoriser un polynome à coefficients entiers est donnée dans le cas du corps Zp mais ce résultat est valable dans un corps fini quelconque , les démonstrations restant inchangées.

3.1 Algèbre de Berlekamp.

Matrice Q de Berlekamp : C'est la matrice (n,n) dont la ligne i vaut $x^{p(i-1)}$ modulo f si f, le polynome à factoriser, est de degré n.

On note Zp[x]/(f) le quotient de l'anneau Zp[x] par l'idéal engendré par f, c'est l'ensemble des classes de résidus modulo f.

Algèbre de Berlekamp : C'est l'algèbre formée des éléments g de Zp[x] tels que : $g^p - g = 0$ mod f.

Proposition : L'algèbre de Berlekamp est l'espace nul de l'application linéaire l : $x \to x^p - x$, dont la matrice est Q-I, I étant la matrice identité (n,n).
Preuve : L'application $x \to x^p$ est linéaire car, dans Zp,

$$(kx)^p = k^p x^p = k\ x^p$$

$$(x + y)^p = x^p + y^p$$

On en déduit le résultat : $g(x^p) = (g(x))^p$ dans $Zp[x]$.

Un élément de l'algèbre B vérifie donc l'équation :

$$(g(x))p - g(x) = 0 \bmod f.$$

Donc B estl'espace nul de l'application linéaire l-i dont la matrice dans Zp x /(f) est Q - I, quand on identifie un polynome à un vecteur.

Exemple : $f = x^4 + x^3 + x^2 + x + 1$ dans $Z_2[x]$.

$1 = 1 \bmod f$

$x^2 = x^2 \bmod f$

$x^4 = x^3 + x^2 + x + 1 \bmod f$ donc $Q = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix}$

$x^6 = x \bmod f$

Les éléments de B sont les polynomes $a + bx + cx^2 + dx^3$, tels que :

$$(Q - I) \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = 0$$

3.2 Factorisation partielle

La connaissance d'une base b1,...,br de l'espace nul de (Q-I) permet d'obtenir une factorisation partielle de f grace au théorème suivant :

Théorème : $f(x) = \prod_{s \in Zp} PGCD(f(x), b(x)-s)$

où b est un élément de l'espace nul de Q-I.

Remarques: - Une base de l'espace nul de Q-I s'obtient par la résolution du système linéaire associé à Q-I.

- Si $b(x) \in Zp$ alors les PGCD se réduisent tous à 1 et on a la factorisation triviale $f(x) = f(x) \prod 1$

- La factorisation obtenue est partielle car les PGCD obtenus peuvent etre réductibles.

- Il peut etre nécessaire de calculer p PGCD d'où

l'inconvénient de la méthode si p est grand.

Preuve du théorème.

$(g(x))^p - g(x) = 0 \bmod f$ entraine que f divise $(g(x))^p - g(x)$

Dans un corps fini , $x^p - x = \prod (x - s)$ donc par substitution ,

$$(g(x))^p - g(x) = \prod_{s \in Zp} (g(x) - s)$$

Réciproquement , il est évident que PGCD(g(x)-s,f(x)) divise f pour tout s de Zp.

Les deux polynomes étant unitaires , ils sont donc égaux.

CQFD.

3.3 - Factorisation complète.

Le résultat précédent ne permet pas de calculer la factorisation complète de f. La proposition suivante permet de conclure.

Proposition : Le nombre de facteurs irréductibles de f est égal à la dimension de l'espace nul de Q-I.

L'algorithme de Berlekamp consiste à prendre un élément b_1 d'une base de Q-I et à calculer les p PGCD de f(x) et de $b_1(x)-s$ pour s parcourant Zp. Si on obtient r facteurs non triviaux, l'algorithme s'arrete, sinon on choisit un autre élément de la base b_2 puis on calcule les PGCD relatifs à b_2. On continue jusqu'à l'obtention des r facteurs.

La démonstration utilise le théorème des restes chinois pour les polynomes . La démonstration de ce théorème se trouve dans Berlekamp (BE 68) p 29.

THéorème des restes chinois : Soient $p_1,..,p_r$ r polynomes irréductibles et $g_1,..,g_r$ r polynomes arbitraires , il existe un polynome h unique modulo le produit des $p_i{}^{e_i}$ tel que :

$$h(x) = g_i \bmod p_i{}^{e_i}.$$

Preuve de la démonstration sur le nombre des facteurs irréductibles:

f est sans carré donc $f = p_1 \ldots p_r$ les p_i étant des polynomes irréductibles distincts.

f divise $\prod (g(x) - s)$ donc pi divise $g(x)-s_i$ pour un certain

si , i.e. $g(x) = s_i \bmod p_i$. Cela est valable pour chaque p_i.

Donc pour tout r uplet $(s_1,\ldots,s_r)$ il existe un unique g modulo le produit des p_i, donc modulo f, tel que $g = s_i \bmod p_i$. Il y a p^r r-uplets possibles , donc p^r solutions g de l'équation $g(x)^p - g(x) = 0 \bmod f$.

Mais l'espace nul de Q-I, de dimension k, qui est l'ensemble des solutions de cette équation a p^k éléments . Les deux ensembles étant identiques k=r .
CQFD.

Cette démonstration reste valable si f possède des facteurs multiples.

3.4 Algorithme complet de factorisation.

L' algorithme de factorisation est le suivant:

Entrée : f un polynome primitif sans carré, unitaire, de degré n.

Sortie : Liste Q1,..,Qr des facteurs unitaires de f.

(1) $h_1 = x^p$ modulo f , $L = \emptyset$
(2) (Calcul de la matrice Q)
Pour i=1,...,n-1 Faire
début
$h_i = h_{i-1}.h_1 \bmod f$
fin
(3) (Calcul de la base)
Calculer une base $H = b_1,\ldots,b_r$ de l'espace nul de Q-I.
(4) (f irréductible) Si r = 1 alors $L = \{f\}$, retour.
(5) $L1 = f$, k=1 , i=1
(6) Tant que $L1 \neq \emptyset$ Faire

début
(6.1) $H = H - \{b_i\}$, A_1 = premier élément de L_1 , $L_1 = L_1 - \{A1\}$
(6.2) Tant que $j < p$ Faire

début
(6.21) $C = PGCD(A_1, b_i)$
(6.22) Si $deg(C) > 0$ alors $L = L \cup \{C\}$, $A_1 = A_1/C$, k=k+1
(6.23) Si k = r alors $L = L \cup L_1 \cup \{A_1\}$, retour.
(6.24) $b_i = b_i + j$, j=j+1
fin

(6.3) i=i+1 , $L_1 = L_1 \cup \{A_1\}$

fin
fin

Les étapes 6.21 à 6.3 représentent les calculs des PGCD entre

f et b_i - s avec b_i un élément de la base choisie de l'espace nul de $Q-I^i$, s étant un élément de Zp.

3.5 Complexité de l'algorithme

On a vu en 2.4 que le cout de construction de la matrice Q est :

$O(n^2L^2(p)\ (n+Logp))$

Le calcul d'une base de l'espace nul de Q-I se fait à l'aide de la méthode de Gauss et coute :

$O(n^3L^2(p))$

Les calculs de PGCD coutent : $O(prn^2L^2(p))$, si r est le nombre de facteurs irréductibles de f.

Le cout global de l'algorithme est donc :

$O(\ n^3L^2(p) + n^2L^2(p)Logp + prn^2L^2(p))$ si n est le degré du polynome à factoriser et r le nombre de facteurs , avec utilisation des méthodes classiques de multiplication.

4 - METHODE PROBABILISTE DE CAMION-LAZARD

4.1 Généralités

L'algorithme de Berlekamp est très efficace quand le nombre premier p est petit, en particulier dans ce cas l'arithmétique de Zp peut se faire à l'aide de tables , et le calcul de l'inverse d'un élément se fait par simple lecture. Mais quand la taille de p augmente cela n'est plus possible , surtout la complexité de l'algorithme étant linéaire en p, le temps de calcul devient prohibitif. Certaines améliorations proposée pour résoudre ce problème ont ete proposées,sans résultats (BE 70). Des algorithmes probabilistes ont alors été mis au point pour résoudre ce problème.

Définition : Un algorithme probabiliste est un algorithme dans lequel une des variables au moins prend une valeur déterminée par un choix aléatoire.

Avec de tels algorithmes il n'est plus possible de parler de complexité dans le sens habituel des algorithmes déterministes , on parle alors de complexité moyenne , qui estime le temps moyen nécessaire au déroulement de l'algorithme.

4.2 Présentation de l'algorithme

Nous présentons l'algorithme de Camion (CA 82) , perfectionné par Lazard (LA 82) dont l'idée a été proposée par Cantor et Zassennauss (CA ZA 81).

Nous présentons l'algorithme de Camion (CA 82) , perfectionné par Lazard (LA 82).

Soit f un polynome sans carré, égal au produit de ses facteurs irréductibles $p_1,\ldots,p_r$.

Zp[x]/(f) est l'ensemble des classes de résidu modulo f , c'est une algèbre réduite isomorphe au produit
$Zp[x]/(p_1) \ldots Zp[x]/(p_r)$

Définition : Un élément u de Zp[x]/(f) est appelé idempotent ssi $u^2 = u$ modulo f.

Il y a un lien étroit entre les idempotents et les facteurs de f et l'idée qui sous-tend l'algorithme est de calculer une décomposition orthogonale des idempotents.

Algorithme de Camion-Lazard

Entrée : f un polynome unitaire sans carré de degré n , p un nombre premier impair.

Sortie : L liste des facteurs irréductibles de f

(1) $h_1 = x^p \bmod f$
(2) Pour i=1,...,n-1 Faire
début
(2.1) $h_i = h_{i-1} \cdot h_1$
fin
(3) Calculer une base de l'espace nul de Q-I
(4) Si r = 1 alors L = {f} , retour
(5) S = {1}
(6) Tant que card(S)<r Faire

début
(6.1) $t = t_1 b_1 + \ldots + t_r b_r$ les t_i sont des éléments aléatoires de Zp
(6.2) $u = t^{(p-1)/2} \cdot (t^{(p-1)/2} + 1)/2 \bmod f$
(6.3) Pour tout s dans S Faire

début
(6.3.1) w = us mod f
(6.3.2) Si w = s et w = 0 alors $S = S - \{s\} \cup \{w, w-s\}$
fin

fin

(7) L = PGCD(f,s-1) pour tous les s de S
fin

Les étapes (1) , (2) ,(3) sont les memes que celles de l'algorithme de Berlekamp.

4.3 Preuve de l'algorithme de Camion-Lazard

Il est nécessaire de rappeler certains résultats sur les idempotents de $A = Zp[x]/(f)$. Ces résultats sont démontrés dans Camion (CA 83).

Proposition : A est somme directe des idéaux (e_i), où (e_i) est l'idéal engendré par l'élément e_i, avec de plus:

$$1 = \sum e_i \quad , \quad e_i e_j = \begin{cases} 0 \text{ si } i \neq j \\ 1 \text{ si } i = j \end{cases}$$

Les e_i sont appelés éléments primitifs de l'algèbre A. On a donc une décomposition de A en une somme directe et orthogonale.

Proposition : l'algèbre de Berlekamp B définie comme espace nul de Q-I se décompose en :

$$B = Zp(e_1) + \ldots + Zp(e_r)$$

De plus les facteurs irréductibles de f sont $p_i = PGCD(f, 1-e_i)$.

La connaissance des idempotents primitifs e_i permet donc de factoriser complètement le polynome f. L'idée de l'algorithme est de décomposer un idempotent u, construit à partir d'éléments de B en idempotents orthogonaux. L'arret se produit quand on a obtenu r idempotents orthogonaux qui sont les idempotents primitifs.

Preuve de l'algorithme de Camion-Lazard

$Zp[x]/(f)$ est somme directe des $Zp[x]/(p_i)$ car f est sans carré.

Si t est un élément de B alors $u = t^{(p-1)/2}(t^{(p-1)/2} + 1)/2$ est un idempotent. En effet modulo p_i, $t^{(p-1)/2} = z_i$ un élément de Zp tel que $z_i^2 = 1$ donc $z_i = +1$ ou -1. Par suite $u^2 = u \bmod p_i$ pour tout les p_i, donc $u^2 = u$.

Montrons que S est formé d'idempotents orthogonaux. A l'initialisation $S = 1$, la propriété est vraie.

Supposons que cela est vrai à une étape de l'algorithme. On rajoute les éléments w et w-s.

w est un idempotent car $w^2 = (us)^2 = u^2 s^2 = w$

w-s est un idempotent car $(s-w)^2 = w^2 + s^2 - 2ws = w + s - 2us = s - w$

w et w-s sont orthogonaux car $w(w-s) = w^2 - ws = w - w = 0$.

Pour tout s' de S distinct de s , $s's = 0$ donc $s'w = uss' = 0$ et $(w-s)s' = ws - ss' = 0$.

Donc $S = S - s \cup w, s-w$ est formé d'idempotents orthogonaux quand on écarte les w égaux à s ou les w nuls.

A chaque étape la somme des éléments de S vaut 1. Donc àla fin de la boucle 5 S est un ensemble de r idempotents orthogonaux dont la somme vaut 1 , S est donc l'ensemble des idempotents primitifs , d'où le résultat $p_i = PGCD(f, 1-s_i)$.

CQFD

4.4 Variantes et améliorations

- Utilisation de la factorisation en produit d'irréductibles de méme degré.

On se ramène à un polynome dont tous les facteurs irréductibles ont meme degré en utilisant l'algorithme de la partie 2. Dans ce cas, à l'étape 6.1, au lieu de choisir un polynome $t=t_1b_1+\ldots+t_nb_n$ on choisit un polynome $t=t_0+t_1x+\ldots+t_{n-1}x^{n-1}$.

La preuve de l'algorithme reste pratiquement inchangée , car l'élément u calculé en 6.2 est un idempotent parce que Zp x /(f) est isomorphe àla somme directe de corps à p éléments. Cette dernière propriété vient du fait que tous les facteurs irréductibles de f ont meme degré.

- Amélioration pour les nombres premiers de la forme 4k+1.

L'idée de l'algorithme de Camion-Lazard est de casser un idempotent en deux idempotents orthogonaux. Quand p est de la forme 4k+1 on va pouvoir casser un idempotents en quatre idempotents au lieu de deux, d'ou un gain sensible en rapidité.

La proposition suivante permet l'amélioration souhaitée due à Waezi (WA 83)

Proposition : Soit t un élément de l'espace de Berlekamp, i un élément de Zp tel que $i^2=-1$, alors si p est de la forme 4k+1 , si $u = t^{(p-1)/4}$, on pose $u_1=(1+u+u^2+u^3)/4$, $u_2=(1-u+u^2-u^3)/4$, $u_3=(1-iu-u^2+iu^3)/4$. Alors les u_j sont des idempotents deux à deux orthogonaux.

Cette proposition découle du fait que $u^4=1$ et donc que u vaut 1,-1,i,-i. Il est alors facile de voir que les u_j sont des idempotents deux à deux orthogonaux.

Calcul d'un élément i racine de -1 : La loi de réciprocité quadratique permet d'affirmer que si a est dans Zp alors la probabilité pour que l'élément $u = a^{(p-1)/4}$ soit une racine de -1 vaut 1/2.

L'algorithme modifié consiste à calculer à l'étape 6.2 un élément $u = t^{(p-1)/4} \bmod f$ puis les u_j correspondants comme

ci-dessus. Ensuite pour chacun de ces u_j on remplacera un élément s de S par $u_j s$ et $s-u_j s$,si ces deux éléments sont non nuls et distincts de s.

4.5 Complexité de l'algorithme

Nous donnons la complexité de l'algorithme originel donné en 4.3.

Il faut pouvoir estimer le nombre de passages à travers la boucle 6 , c'est à dire le nombre de choix d'éléments aléatoires de B nécessaires avant d'obtenir un partition de S en idempotents primitifs.

Proposition : Soit r le nombre de facteurs de f , alors I(r) le nombre moyen d'itération pour décomposer f en idempotents primitifs vérifie $I(r) < c \, Logr$ ou c est une constante positive.

Cette proposition est due à Lazard (LA 82) ,qui conjecture que c=2,2 pour des nombres premiers supérieurs à 5.

La complexité moyenne est évaluée comme suit:

- Le cout des étapes 1 à 3 a déjà été calculé pour l'algorithme de Berlekamp.

- Les étapes 6.2 à 7 coutent :

  M(n)Logp pour l'étape 6.2 en utilisant la matrice Q

  M(n) pour le calcul de us en 6.3.1.

  GCD(n) pour le PGCD de f et de s-1 en 7

  Les étapes 6.2 et 6.3.1 sont faites I(r) fois , et l'étape 7 r fois.

Le cout de l'algorithme de Camion-Lazard est donc de:

$O(n^3 + rn^2Logr + n^2LogrLogp)L^2(p))$ pour les méthodes classiques

$O(n^3 + nLognLogrLogp + rnLogp)L^2(p))$ avec la FFT

La complexité de l'algorithme est donc en logp au lieu d'etre linéaire en p comme pour l'algorithme de Berlekamp. Donc on peut envisager l'utilisation effective de cet algorithme pour des p grands.

5 COMPARAISONS ET PERSPECTIVES.

5.1 Algorithme de Cantor-Zassenhauss.

Cet algorithme a été donné par Cantor et Zassenhauss (CA ZA 81) antérieurement à l'algorithme de Camion-Lazard. L'idée de base est proche mais différentes versions sont possibles, la plus simple est celle donnée par Knuth (KN 81),& 4.6.2.Son importance est liée à l'introduction des méthodes probabilistes par Rabin (RA 80) pour factoriser des polynomes définis sur un corps fini. Elle constitue une importante amélioration des méthodes de Rabin et elle est très simple à implémenter, d'où son interet pour comparer diverses méthodes de factorisation (CA LO 82).

L'idée de cette méthode est que si t est un polynome aléatoire de degré inférieur à 2d, si f est un produit de polynomes irréductibles de degré d , alors

$$f=PGCD(f,t))(PGCD(f,t^{(p^d-1)/2}-1)(PGCD(f,t(^{p^d-1)/2}+1))$$

est une factorisation non triviale de f avec une probabilité de 1/2 à peu près. (en fait $1/2 - 1/(2p)^d$.)

En moyenne on effectue 2r choix aléatoires de polynomes t et la complexité de l'algorithme est :

$O(rn^3Log(p)L^2p)$ pour les méthodes classiques.

Mais il est nécessaire de factoriser f en produit de facteurs irréductibles de meme degré avant d'utiliser cet algorithme.

5.2 Comparaison

J.Calmet et R.Loos (CA LO 82) ont comparé l'algorithme de Berlekamp et celui de Cantor-Zassenhauss ,l'algorithme de Camion-Lazard n'étant pas disponible à l'époque de leurs travaux.

Dans tous les cas la factorisation partielle en produit d'irréductibles de meme degré a été utilisée.

Pour de petits nombre premiers les deux algorithmes étaient sensiblement équivalents. Mais la méthode de Cantor-Zassenhauss calcule des termes du type $t^{(p^d-1/2)}$ ce qui la rend peu efficace dès que p est grand.

Cependant il est apparu que c'est la factorisation partielle qui est la partie la plus couteuse, spécialement la construction de la matrice Q. Un des exemple proposé par Calmet-Loos donne pour p=45035996273711451 un temps de 154s pour la factorisation partielle dont 147s pour construire Q , et un temps de 79s pour la factorisation des facteurs partiels.

5.3 Perspectives

L'algorithme de Berlekamp reste le plus efficace pour factoriser modulo un petit nombre premier. Par contre dès qu'on veut utiliser des nombres premiers grands, les algorithmes probabilistes s'imposent. Parmi ceux-ci l'algorithme de Camion-Lazard semble le plus recommandable à cause du cout moindre nécessité par les exponentiations de $t^{(p-1)/2}$ au lieu de $t^{(p^d-1)/2}$ pour l'algorithme de Cantor-Zassenhauss.

Néanmoins le problème reste posé de la construction de la matrice Q en un temps plus court ou bien de la possibilité de ne pas l'utiliser. Il s'agit du problème de la réduction du nombre d'exponentiations qui reste ouvert et deviendra sans doute le point de recherche principal dans le domaine de la factorisation sur un corps fini.

Un autre point important est que l'implémentation de l'algorithme de Berlekamp a été constamment améliorée au court des dernières années. Les algorithmes probabilistes n'ont été implémentés que récemment et il est prévisible que leur programmation progressera de façon similaire.

CHAPITRE 2 : FACTORISATION DES POLYNOMES A UNE VARIABLE A COEFFICIENTS ENTIERS

Ce chapitre aborde le problème de la factorisation d'un polynome à une variable à coefficients entiers. Une première partie traite des la simplification qui rend un polynome sans carré et des bornes utilisées dans la factorisation. Les méthodes utilisées sont décrites avec un paragraphe spécial pour la méthode classique. Les problèmes rencontrés sont décrits , en particulier celui des facteurs parasites. L'algorithme de Lenstra qui résoud ce problème est donné. Des résultats de mesures de temps de calcul permettent d'apprecier l'efficacité des algorithmes. Les perspectives du problème sont rapidement évoquées.

1 FACTORISATION SANS CARRE ET BORNES UTILISEES

1.1 Factorisation sans carré

Avant de factoriser le polynome P , il est important de chercher à le simplifier si cela est possible.

On va d'abord rendre le polynome primitif. Pour cela on calcule le PGCD de l'ensemble des coefficients du polynome. Si ce PGCD noté c , n'est pas 1 , on obtient une première factorisation: $P = c.P/c$.

Ensuite on cherche si P possède des facteurs multiples , afin de se ramener à un polynome qui est produit de facteurs irréductibles premiers deux à deux. L'idée de base pour effectuer une telle réduction est que si P a un facteur multiple Qi à l'ordre n_i, alors sa dérivée P'=DP/Dx possède ce facteur à l'ordre n_i-1. Par suite le PGCD de P et de P' est le produit de tous les Qi à la puissance n_i-1, et le quotient de P par ce PGCD est sans carré.

Il est donc possible de factoriser ce quotient puis de faire les divisions de P par les facteurs obtenus jusqu'à trouver la puissance des facteurs irréductibles de P. Mais il existe des algorithmes plus performants. Les premiers sont dus à Horowitz (HO 71) puis ils ont été améliorés; on donne ici l'algorithme de Yun (YU 76).

Algorithme de Yun pour la factorisation sans carré.

on pose $P = Q_1^1 Q_2^2 \ldots Q_k^k$ avec Q_i polynomes premiers entre eux deux à deux ou bien $Q_i = 1$.

$$P'=DP/Dx = Q_2^1 Q_3^2 \ldots Q_k^{k-1} \left(\sum_i Q_i' \prod_{j\neq i} Q_j \right)$$

on pose $D=PGCD(P,P')=Q_2^1 Q_3^2 \ldots Q_k^{k-1}$

Proposition : $PGCD(P'/D - (P/D)', P/D) = Q_1$

Cette proposition est la base de l'algorithme et montre comment les Q_i vont apparaitre comme PGCD de polynomes construits sur le modèle de ceux qui apparaissent ici.

Démonstration : $(P/D)' = \sum_i Q_i' \prod_{j\neq i} Q_j$

d'où $P'/D - (P/D)' = \sum_{j \neq i} (i-1) Q_i' \prod Q_j$ car $P/D = Q_1 \ldots Q_k$

Par suite Q_1 estle seul facteur de P/D qui divise $P'/D - (P/D)'$. Donc $PGCD(P'/D - (P/D)', P/D) = Q_1$.

CQFD

Algorithme de Yun.

Entrée : P polynome à coefficients entiers.

Sortie : Liste L = $((1,Q_1),\ldots,(k,Q_k))$ où Q_i est le facteur multiple de degré k de P ou $Q_i = 1$

(1) D = PGCD(P,P')
(2) P/D = C1
(3) P'/D = D1
(4) Tant que Ci ≠ 1 Faire

(5) début
Qi = PGCD(Ci,Di - Ci')
(6) Ci+1 = Ci/Qi
(7) Di+1 = (Di - Ci')/Qi , i=i+1
fin

(8) Sortir la liste ((1,Q1),...,(k,Qk))
fin

Preuve de la validité de l'algorithme.

On va montrer qu'à l'étape i , $C_i = \prod_{j \geq i} Q_j$ et $D_i = H_i/E_i$ avec $H_i = \prod_{j \geq i} Q_j^{j-i+1}$ et $E_i = PGCD(H_i, H_i')$ par réccurence.

Pour i = 1 le résultat est donné par la proposition précédente.

Supposons le résultat vrai pour i.

Alors Q_i est le PGCD de C_i et de $D_i - C_i'$ d'après la même proposition.

Donc $C_{i+1} = C_i/Q_i = \prod_{j \geq i+1} Q_j$.

$D_i - C_i' = Q_i \sum_{j \geq i+1} (j-1) Q_j' \prod_{k \neq j, k \geq i+1} Q_k$

$Di+1 = \left(\prod_{j \geq i+1} Q_j^{j-1} \right)' / \left(\prod_{j \geq i+1} Q_j^{j-1} \right)$ d'où $D_{i+1} = (H_{i+1})'/E_{i+1}$

CQFD

Exemple: Nous donnons les étapes les plus significatives de l'algorithme de Yun sur l'exemple suivant.

$$P(x) = x^9+3x^8+2x^7-2x^6+4x^5-4x^4-2x^3+2x^2+3x+1$$

$C_1 = x^4-1$ et $D_1 = 9x^3-3x^2+5x-3$ avec $Q_1 = x^2+1$

$C_2 = x^2-1$ et $D_2 = 5x - 3$ avec $Q_2 = x-1$

$C_3 = x+1$ et $D_3 = 3$ avec $Q_3 = 1$

$C_4 = x+1$ et $D_4 = 2$ avec $Q_4 = 1$

$C_5 = x+1$ et $D_5 = 1$ avec $Q_5 = x+1$

$C_6 = 1$

Résultat : $P(x) = (x^2+1)(x-1)^2(x+1)^5$

Complexité de l'algorithme de Yun .

On pose $n_i = \deg(Q_i)$ d'où $n = \sum n_i$.
$c_i = \deg(C_i)$ d'où $c_i = \sum_{j=i,..,k} n_i$

De plus comme $D_i = H_i'/E_i$ on a $\deg(D_i) = c_i-1$

Cout des étapes : étape 1 $O(\log n\, M(n))$ qui est le cout du PGCD.
étape 2 et 3 : $O(M(n))$
boucle : étape 5 cout $O(\log c_i\, M(c_i))$t le cout du PGCD
étape 6 cout $O(M(c_i))$
étape 7 cout $O(M(c_i))$

d'où un cout total de $O(\log c_i\, M(c_i))^i$ pour les trois étapes soit un cout en $O(\log n\, M(n))$ pour l'ensemble.

La complexité de l'algorithme de Yun est $O(\log n\, M(n))$ soit un cout de $O(n^2 \log n)$ pour les méthodes classiques.

1.2 Bornes sur les facteurs d'un polynome

Les méthodes actuelles de factorisation ne sont valables que parce qu'il est possible de majorer la taille de tout coefficient de tout diviseur éventuel d'un polynome donné. Si on connait les facteurs modulo m d'un polynome , on peut donc en déduire les facteurs sur Z[x] du polynome si m est assez grand.

Bornes actuelles : $M = |a_n| . 2^n . \sum |a_i|$ (GE 60)

ou $M = |a_n| . 2^n . ((\sum |a_i|^2)^{1/2} + 1)$ (MI 74)

si le polynome à factoriser s'écrit $\sum_{i=0,..,n} a_i x^i$

Plus précisément Mignotte a montré que si Q est un facteur de degré i , alors les coefficients de Q sont majorés en valeur absolue par M_i , $Mi = |a_n| . 2^i . ((\sum a_i^2)^{1/2} + 1)$.

La deuxième borne est meilleure car $(\sum |a_i|^2) < (\sum |a_i|)^2$ quand les a_i ne sont pas tous nuls.

De plus cette deuxième borne est optimale c'est à dire qu'il existe des polynomes pour lesquels elle est atteinte par un coefficient d'un facteur (MI 80).

Par conséquent on va calculer les facteurs d'un polynome jusqu'à un module m > 2M pour etre sur d'obtenir une taille suffisante pour les coefficients d'éventuels facteurs. Le terme 2M est utilisé au lieu de M car la représentation des entiers utilisée est celle qui considère les entiers compris entre 0 et m. Pour obtenir les facteurs sur Z on prend la représentation des entiers entre -m/2 et m/2.

exemple : $P(x) = 15x^2 - 47x + 28$

$M = 464$

$Q(x) = 5x + 1327$ modulo 11^3

On prend la forme de Q(x) dont les coefficients sont dans]-665,665] ce qui donne Q(x) = 5x - 4 qui divise P(x).

2 METHODES DE FACTORISATION SUR Z

2.1 Méthode historique (Kronecker , Schubert)

Nous donnons l'approche historique car il est interessant de connaitre l'approche adoptée à l'époque et aussi car c'est la seule connue pour certains domaines comme le corps des complexes C.

L'idée de base est que si Q(x) est un facteur du polynome P(x), alors Q(a) divise P(a). Donc on choisit, pour tout degré

possible l ,l+1 entiers a_i on factorise $P(a_i)$, puis pour tous les choix possibles de facteurs de $P(a_i)$ on construit le polynome qui prend ces valeurs en a_i à l'aide de la formule d'interpolation de Lagrange. Ensuite on teste si le polynome obtenu divise bien P. L'explication complète de la méthode peut se trouver dans le livre de Van der Varden (VDV 53).

Il est clair que la complexité de la méthode est exponentielle en le degré du polynome. De plus il est nécessaire de factoriser des entiers qui peuvent etre grands , ce qui est couteux; et de fait cette méthode s'est révélée impossible à mettre en oeuvre pratiquement.

2.2 Méthodes classiques

La méthode classique de factorisation se déroule en trois étapes.

(1) Factorisation sur Zp [x] : on factorise l'image du polynome par l'homomorphisme naturel entre Z et Zp, où p est un nombre premier bien choisi en général petit.

(2) Remontée de la factorisation partielle modulo p en une factorisation modulo p^k : on effectue cette remontée jusqu'à ce que p^k soit supérieur à la borne préalablement calculée sur les coefficients de tout diviseur éventuel de P. Cette remontée s'effectue à l'aide du lemme de Hensel ou de méthode p-adiques semblables.

(3) Reconstruction des facteurs de P : à l'aide de la factorisation modulo p^k, on obtient les facteurs de P en recombinant les facteurs modulo p^k puis en testant si les facteurs obtenus divisent P. L'algorithme se termine soit quand on a une factorisation complète de P soit quand toutes les conbinaisons possibles sont épuisées.

Un nombre premier est "bien choisi" si modulo p le polynome reste sans carré et si p ne divise pas le coefficient de tête de P. Ces conditions sont équivalentes au fait que P ne divise pas le résultant de P et de sa dérivée.

Les méthodes de factorisation modulo p ont été abordées au chapitre 1. Avant d'aborder en détail la méthode classique décrite ci-dessus , nous donnons d'autres méthodes de factorisation sur Z [x].

2.3 Méthodes des restes chinois.

Elle diffère de la précédente en ce sens qu'au lieu de calculer une factorisation modulo p puis de la remonter, elle calcule plusieures factorisations modulo $p_1, p_2, \ldots, p_k$ où les p_i sont des nombres premiers. La factorisation sur $Z[x]$ est calculée à l'aide de ces factorisations en utilisant le théorème des restes chinois.

Le théorème des restes chinois , ainsi nommé parce qu'il était connu d'astronomes chinois qui l'utilisaient pour calculer des dates d'eclipse, a été donné au chapitre 1 nous en donnons un des énoncés possibles.

Théorème des restes chinois : Soient k nombres premiers distincts, soient k polynômes $f_1, f_2, \ldots, f_k$ de meme degré appartenant respectivement à $Zp_1[x], \ldots, Zp_k[x]$, alors il existe un polynome g unique modulo le produit des pi tel que :

$$g = fi \bmod pi$$
$$\deg(g) = \deg(fi)$$

La preuve de cet énoncé est une copie de la preuve du théorème classique des restes chinois connu pour les entiers. On peut la trouver dans Berlekamp (BE 68) .

Cette méthode de factorisation suggérée par Knuth (KN 81) n'est pas utilisée actuellement . En effet le produit p1...pk doit etre supérieur à la borne M qui peut etre très grande, donc il faut pouvoir disposer d'une grande liste de nombre premiers et pouvoir factoriser de manière efficace modulo ces nombres premiers. Or les algorithmes déterministes de factorisation utilisés actuellement ont un cout linéaire en p ce qui peut etre prohibitif dans le cas présent quand les p_i ne sont plus petits.

2.4 Méthodes de factorisation modulo un grand nombre premier

Ces méthodes sont apparues avec les algorithmes probabilistes de factorisation dans Zp[x], notament ceux dus à Rabin (RA 80) et Cantor et Zassenhauss (CA ZA 80) et avec les tests probabilistes de Rabin (RA 80) pour savoir si un entier est composé ou non.

Actuellement on sait tester si de grands nombres sont premiers , en particulier un algorithme très rapide permet d'affirmer soit qu'un nombre est composé , soit que la probabilité pour qu'il le soit est inférieure à un nombre fixé à l'avance aussi petit qu'on le désire. Les nombres qui ne sont pas trouvés composés sont appelés pseudo premiers car on ne peut pas affirmer avec certitude qu'ils sont premiers ,mais la probabilité pour qu'ils ne le soient pas est très faible.

La méthode consiste donc à générer un pseudo premier supérieur en valeur absolue à la borne relative au polynome à factoriser, puis à factoriser le polynome dans le corps fini relatif à ce nombre premier à l'aise d'un algorithme probabiliste. Il suffit ensuite de passer directement à l'étape de reconstruction des facteurs. Donc l'étape 2 des méthodes classiques , basée sur le lemme de Hensel est court-circuitée. Pour que cette méthode soit efficace il faut donc disposer d'algorithmes efficaces pour la factorisation modulo un grand nombre premier et seuls les algorithmes probabilistes répondent à cette exigence pour l'instant.

3 LEMME DE HENSEL ET RECONSTRUCTION DES FACTEURS.

3.1 Lemme de Hensel

Nous donnons ici la forme la plus simple du lemme de Hensel, i.e. la forme série linéaire. Les autres formes possibles font l'objet du chapitre suivant.

Lemme de Hensel : Soit P un polynome de $Z[x]$ sans carré, soit p un nombre premier tel que P modulo p soit sans carré, soit a un facteur unitaire de P modulo p et soit b le cofacteur associé à a, alors pour tout $q = p^k$, il existe a_k et b_k tels que :

$$P = a_k\, b_k \bmod p^k$$

$$a_k \text{ soit unitaire et } a_k = a \bmod p$$

Démonstration . P étant sans carré modulo p le polynome a est premier avec b. Il existe donc deux polynomes u et v tels que : $au + bv = 1 \bmod p$ et $\deg(u) < \deg(a)$ d'après une identité de Bezout.

La démonstration du lemme se fait par réccurence et utilise l'identité de Bezout de l'étape précédente pour calculer les nouveaux facteurs.

Pour k = 1 le résultat est vrai.

Supposons le résultat vrai à l'ordre k.
$P = a_k\ b_k \bmod p^k$ avec a_k unitaire et $a_k = a \bmod p$

Cherchons a_k' et b_k' tels que:

$$P = (a_k + p^k\ a_k')(b_k + p^k\ b_k') \bmod p^{k+1}$$

$$\text{Alors}\quad P - a_k\ b_k = p^k(a_k'b_k + b_k'a_k) + p^{2k}\ a_k'\ b_k' \bmod p^{k+1}$$

$$\text{D'où}\quad (P - a_k\ b_k)/p^k = (a_k'b_k + b_k'a_k) \bmod p$$

Posons S = (P - akbk)/pk , on sait que S est un polynome de Zp[x] Il faut trouver a_k' et b_k' éléments de Zp [x] satisfaisant l'équation précédente.

On connait u et v tels que : au + bv = 1 mod p.
Multiplions cette équation par S alors aSu + bSv = S mod p ce qui donne :

$$a(qb + u) + br = S \bmod p \text{ et } \deg(r) < \deg(a) \bmod p$$

si la division de Sv par a a pour reste r et pour quotient q. Cette division est possible car a est unitaire et de plus Zp est un corps.

On pose alors $a_k' = r$ et $b_k' = qb + u$.

Alors a_{k+1} et b_{k+1} construits comme indiqués satisfont bien à la condition voulue. De plus on a par construction $a_{k+1} = a \bmod p$ et $b_{k+1} = b \bmod p$.

CQFD

3.2 Reconstruction des facteurs

Comme on l'a vu le lemme de Hensel permet de remonter un facteur unitaire de P modulo p en un facteur unitaire modulo p^k. Appliquée à tous les facteurs , cette méthode permet d'obtenir tous les facteurs unitaires de P modulo p^k. Il faut pouvoir en déduire les facteurs de P sur Z[x].

Deux problèmes surgissent alors, celui du coefficient de tete et ceui des facteurs étrangers ou parasites. Ils seront étudiés en détails au paragraphe suivant, nous ne donnons ici que la méthode classique de résolution.

- Problème du coefficient de tête : Si le polynôme de départ n'est pas unitaire, il y a problème car les facteurs remontés sont unitaires. Soit Q' un facteur de P modulo p^k qui est l'image d'un facteur Q de P au coefficient de tête près. Comme Q' est unitaire $Q = a_q Q' \bmod q$ avec $q = p^k$ si a_q est le coefficient de tete de Q.

Mais il n'est pas possible de déterminer a_q, tout ce qu'on connait de lui est qu'il divise a_n le coefficient de tete de P.On sait que P est primitif donc Q est primitif.

Il y a deux méthodes pour trouver le facteur Q.

Soit on considère le polynome primitif déduit de $a_n Q$, soit on regarde si $a_n Q$ divise $a_n P$ puis on calcule le polynome primitif. Un exemple illustre l'une et l'autre méthodes.

$$P(x) = 30x^3 + 83x^2 - 162x - 455$$

Le module est $p^k = 11^4 = 14641$, et on obtient un facteur unitaire modulo 11^4 qui est $Q'(x) = x + 3416$. Donne-t-il un facteur de P?

1° méthode : $a_n Q'(x) = 30x - 70 \bmod 11^4$ et le polynome primitif qui s'en déduit est $3x - 7$ qui divise P(x).

2° méthode : $a_n Q'(x) = 30x + 102480 = 30x - 70 \bmod 11^4$ qui divise le polynome $a_n P(x) = 900x^3 + 2490x^2 - 4860x - 13650$.

Dans les deux cas le facteur de P(x) égal à $3x - 7$ a été trouvé.

La seconde méthode est avantageuse quand le coefficient de tete est petit , mais peut poser problème quand il est grand . De plus elle ne dispense pas de calculer le polynome primitif quand le reste de la division est nul.

La première méthode peut etre désavantageuse si le facteur Q' ne donne pas de facteur vrai de P car on a calculé inutilement le polynome primitif déduit de $a_n Q'$

- Facteurs parasites : les facteurs unitaires modulo pk ne donnent pas toujours un facteur vrai de P . Mais un facteur de P est nécessairement produit de facteurs modulo pk, au coefficient de tete près. Tant qu'on ne trouve pas de facteur effectif de P sur Z il faut tester tous les choix possibles de facteurs modulo pk de degré inférieur ou égal à n/2 . On se limite à n/2 car un facteur de degré supérieur a un cofacteur dont le degré est inférieur à n/2.

Algorithme de recherche des facteurs de P.

EEtrée : P le polynome primitif sans carré à factoriser et l la liste de ses r facteurs unitaires modulo p_k rangés par degré croissant.

Sortie : Liste L des facteurs de P

(1) k =1
(2) Tant que k < r-1 Faire
(3) Prendre l1 liste extraite de l , à k éléments
(4) Q = produit des éléments de l1
(5) Diviser P par le polynome primitif déduit de a_nQ
(6) Si le reste est nul alors
(6.1) éliminer les éléments de l1 dans l
(6.2) mettre le polynome primitif Q' trouvé dans la liste l
(6.3) remplacer P par P/Q'
(7) Si le reste est nul alors éliminer l1 des listes à k éléments possibles , si les listes à k éléments ne sont pas épuisées aller en 3
(8) k = k+1
(9) Si k = r - 1 alors P est irréductible , rajouter P à l , retour , sinon aller en 3
fin

4 PROBLEMES DE LA METHODE CLASSIQUE

4.1 Coefficient de tete

La forme du lemme de Hensel donnée donne comme résultat les facteurs unitaires de P modulo p^k. La factorisation modulo p ne pose pas de problème quand p est petit car il est choisi de façon à ne pas diviserle coefficient de tete qui est donc inversible dans Zp et son inverse a une taille limitée par p. Mais le fait d'obtenir les facteurs unitaires modulo p^k signifie que les facteurs sont multipliés par l'inverse du coefficient de tete dans Zp^k.Cet inverse existe car p ne divise pas a_n mais il peut etre très grand , meme si a_n est petit.

exemple : $q = 11^4$ et $a_n = 2$ alors $a_n^{-1} = 7321$

Donc les coefficients des facteurs sont considérablement augmentés par rapport au coefficients vrai de P.

La solution serait de précalculer les coefficients de tete des facteurs de P, en supposant qu'il n'y ait pas de facteurs parasites. Mais il n'existe pas de méthode permettant de le

faire. Le seul résultat qui existe est un résultat de Wang (WA 83) qui permet de calculer un rationnel à partir de son image modulaire. Le problème du précalcul des coefficients de tete des facteurs reste ouvert.

4.2 Borne sur les facteurs

La borne qui majore la valeur absolue de tout coefficient de tout facteur de P est souvent beaucoup trop grande.

exemple : $P(x) = 21x^6 - 120x^5 - 189x^4 + 21x^2 - 120x - 189$

$m = 887040$ et la borne vaut 5^9 si $p = 5$

Ainsi on remonte les facteurs de P jusqu'à $5^9 = 1953125$ alors que ces facteurs sont x^4+1 , $7X+9$, $3X-1$.

Pour remédier à la situation , on introduit une borne heuristique m" et on teste les facteurs obtenus modulo p^l dès que p^l est supérieur à m". Très souvent on trouve un ou plusieurs facteur de P et le problème de la factorisation de P se réduit à celle de P/Q si Q est le produit des facteurs trouvés. De plus il arrive fréquemment que le module p^l dépasse la nouvelle borne relative à P/Q ce qui permet de passer directement à l'étape de reconstruction des facteurs.

La borne heuristique la plus correcte intuitivement est $m" = M^{(1/r)}$ où r est le nombre de facteurs modulo p.

Dans l'exemple m" vaut 125 les facteurs de P modulo 125 sont $x^2 + 32$, $x^2 + 93$, $x + 37$ et $x + 118$.

On trouve $21(x + 37) = 3(7x + 9)$ et $7x + 9$ divise P. De meme $3(x + 118) = 3x - 21$ qui divise P. On est ramené à factoriser $x^4 + 1$ dont on connait deux facteurs modulo 125 . La borne relative à $x^4 + 1$ est M = 32 qui est déjà dépassée, donc $x^4 + 1$ est irréductible car ses facteurs modulo 125 ne donnent pas de vrai facteur sur Z[x]. La factorisation de P est complète et il n'a pas été nécéssaire d'aller jusqu'au module $M = 5^9$ le module 5^3 a suffi.

4.3 Facteurs parasites

Quand on passe de Z[x] à Zp[x] , le problème de la factorisa-

tion se complique car un polynome irréductible sur Z[x] peut etre réductible sur Zp[x].

Par exemple $x^2 + 1$ est irréductible sur les entiers mais se scinde en $(x + 1)(x + 1)$ dans $Z_2[x]$.

Si P a r facteurs irréductibles sur Z[x] , il a au moins r facteurs sur Zp[x], mais il peut en avoir plus. Ce problème est connu sous le nom de problème des facteurs parasites ou étrangers "extraneous factors". C'est le problème majeur de la factorisation car il introduit de nombreux calculs supplémentaires . Le plus mauvais cas est celui où le polynome de départ est irréductible et possède n = deg(P) facteurs irréductibles. Dans ce cas l'étape de reconstruction des facteurs impose de tester les 2^n choix possibles de facteurs , donc l'algorithme est exponentiel en le degré du polynome.

La question se pose alors du choix du nombre premier p , peut-etre qu'un autre choix arrangerait la situation ? Il n'en est rien, ce que montre la proposition suivante due à Swinnerton-Dyer.

Proposition : Soit n un entier , p1,...,pn des nombres premiers distincts positifs , alors le polynome dont les racines sont : $\pm\sqrt{p_1} \pm \sqrt{p_2} \pm \ldots \pm \sqrt{p_n}$ est un polynome à coefficients entiers , irréductible , sur Z qui se décompose en produit de polynomes de degré 2 au plus modulo tout nombre premier p.

Démonstration : Elle est peu classique , aussi nous la présentons. Cette approche est tirée de Kaltofen (KA 82)

Elle se fait par récurrence sur n .

Soit f_k le polynome correspondant à $p_1,\ldots,p_k$.
Soit $L_k = Q(p_1,\ldots,p_k)$ on va montrer que f_k est dans Z[x] et que $[L_k : Q] = 2^k$, et que $\sqrt{p_1} + \ldots + \sqrt{p_n}$ est un élément primitif de L_k . Pour toutes les notations utilisées sur la théorie de Galois se reporter à Van der Varden (VDV 53)

Pour k = 1 le théorème est vrai.

Par construction , $f_k(x) = f_{k-1}(x + \sqrt{p_k})\, f_{k-1}(x - \sqrt{p_k})$ donc f_k est dans $Z[\sqrt{p_k}, x]$ d'après l'hypothèse de récurrence. Mais les coefficients de f_k sont des fonctions symétriques en $\sqrt{p_k}$ et $-\sqrt{p_k}$ donc ce sont des entiers.

On peut montrer que $f_k(x)$ est égal au résultant par rapport à y de $f_{k-1}(x-y)$ et $x^2 - p_n$.

Par hypothèse $[L_{k-1} : Q] = 2^{k-1}$, une base de L_{k-1} est alors la

suivante :
1 et les éléments $\sqrt{p_{i_1}\ldots p_{i_j}}$ pour tous les j-uplets tels que $j=1,\ldots,l$ avec $1 \leq l \leq k-1$, les i_j étant strictement croissants.

Montrons par récurrence que pk n'est pas dans L_{k-1} , alors il sera de degré 2 sur Lk-1 et on aura :

$$[L_k : Q] = [L_k : L_{k-1}][L_{k-1} : Q] = 2^k$$

Si $\sqrt{p_k}$ est dans L_{k-1} il est combinaison linéaire d'éléments de la base de L_{k-1} donnée. Il existe au moins deux composantes de p_k non nulles , on peut supposer, sans restreindre la généralité du cas étudié que p_{k-1} intervient dans un des deux éléments de base correspondants.

$\sqrt{p_k} = s_1 + \sqrt{p_{k-1}}\, s_2$ avec s_i dans L_{k-2} , on en déduit :

$p_k = {s_1}^2 + {s_2}^2 p_{k-1} + 2s_1 s_2 \sqrt{p_{k-1}}$ et donc une expression de $\sqrt{p_{k-1}}$

$\sqrt{p_{k-1}} = (p_k - {s_1}^2 - {s_2}^2 p_{k-1})/s_1 s_2$ donc $\sqrt{p_{k-1}} \in L_{k-2}$ ce qui est contraire à l'hypothèse de récurrence.

Donc $[L_k : Q] = 2^k$

Montrons que $L_k = Q(s)$ avec $s = \sqrt{p_1} + \sqrt{p_2} + \ldots + \sqrt{p_k}$. Pour cela on considère les polynomes de $Q(s)[x]$ qui sont :

$$g(x) = f_{k-1}(\sqrt{p_1} + \ldots + \sqrt{p_{k-1}} + (\sqrt{p_k} - x)) \quad \text{et} \quad h(x) = x^2 - p_k$$

On va montrer que leur PGCD est $x - \sqrt{p_k}$.

Ce polynome divise bien h et g par construction. Comme h(x) est égal à $(x - \sqrt{p_k})(x + \sqrt{p_k})$, il suffit de montrer que $x + \sqrt{p_k}$ ne divise pas g.

S'il divisait g alors $\sqrt{p_1} + \sqrt{p_2} + \ldots + 2\sqrt{p_k}$ serait une racine de g , donc un élément z de L_{k-1} , par suite :
$\sqrt{p_k} = 1/2(z - (\sqrt{p_1} + \ldots + \sqrt{p_k}))$ serait dans L_{k-1} et on a démontré que ce n'était pas possible.

Donc le PGCD de g et h est $x - \sqrt{p_k} \in Q(s)[x]$ et donc

$Q(s) = Q(\sqrt{p_1} + \sqrt{p_2} + \ldots + \sqrt{p_{k-1}}, \sqrt{p_k}) = Q(\sqrt{p_1} + \ldots + \sqrt{p_k})$ car on a vu que $L_k = Q(\sqrt{p_1} + \ldots + \sqrt{p_{k-1}})$.

On peut alors montrer que fk est irréductible sur Z ou sur Q , ce qui est équivalent.

Le polynome fk à coefficients entiers , unitaire de degré 2^k , admet pour racine $\sqrt{p_1} + \sqrt{p_2} + \ldots + \sqrt{p_k}$ qui est algébrique de degré

2^k sur Q , donc fk est irréductible.

Il suffit de montrer que le polynome se factorise modulo tout nombre premier q pour terminer la démonstration du théorème.

Modulo q , p_i est dans $F_q 2$, donc toutes les racines de f sont dans $F_q 2$. Si f possédait un facteur de degré m 2 , ses racines engendreraient le corps $F_q m$, donc ne seraient pas éléments du corps $F_q 2$.

CQFD

Complexité.

La complexité de l'algorithme de Berlekamp est $O(n^3 + prn^3)$, si n est le degré du polynome à factoriser , p le nombre premier utlisé et r le nombre de facteurs modulo p.

On verra au chapitre suivant que la complexité de l'algorithme de remontée est O(n4 L2(d)) si d majore la valeur absolue des coefficients de P.

L'étape de reconstruction a une complexité $O(2^r n^2 L^2(d))$.

La complexité de l'algorithme de Berlekamp-Hensel est exponentielle en le degré du polynome à factoriser.

5 TRAITEMENT DU PROBLEME DES FACTEURS PARASITES

5.1 Méthode empirique

Il s'agit des méthodes utilisées par les systèmes de calculs actuels pour résoudre le problème crucial des facteurs parasites qui peut rendre l'algorithme classique exponentiel.

En général un polynome possède un ou plusieurs facteurs parasites. Mais en combinant plusieures factorisations modulaires, il estpossible d'écarter certains de ces facteurs.

exemple : $u(x) = x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5$

Modulo 13 , $u(x) = (x^4 + 2x^3 + 3x^2 + 4x + 6)$

$(x^3 + 8x^2 + 4x + 12)(x + 13)$

Modulo 2 , $u(x) = (x^6 + x^5 + x^4 + x + 1)(x^2 + x + 1)$

Le polynome u a un facteur de degré 6 modulo 2 , et des facteurs de degré 4 , 3 , 1 modulo 13. Le facteur de degré 6 ne peut etre produit de polynomes dont les degrés valent 4 , 3 , 1. Donc ce polynome est irréductible.

Musser (MU 78) a étudié le nombre moyen de factorisations à effectuer pour conclure à l'irréductibilité d'un polynome aléatoire.

En pratique, on calcule plusieurs factorisations modulaires, on regarde si les degrés sont compatibles et on retient le nombre premier qui donne un nombre de facteurs minimal. Dans ALDES/SAC2 par exemple, on teste 10 petits nombres premiers. Si cette méthode ne permet pas de conclure définitivement à l'élimination des facteurs parasites, elle en élimine un grand nombre.

5.2 Méthode statistique

Si dans la pratique l'apparition de facteurs étrangers se produit fréquemment , en revanche le cas le plus mauvais se produit rarement , et le comportement de l'algorithme semblait polynomial.

Il était intéressant de faire une étude statistique de cet algorithme , ce qu'a fait Mignotte. Pour avoir un exposé précis de cette étude voir Mignotte (MI 80) , car seules les idées essentielles seront décrites ici.

Un polynôme à coefficient entiers est presque toujours irréductible , c'est un résultat de Collins (CO 79).

Ensuite Mignotte montre qu'un polynome de degré n donne 2Log(n) facteurs en moyenne modulo p.

Donc la complexité de la troisième étape , celle de reconstruction devient $O(n^4 L^2(d))$, et par suite la complexité de l'algorithme entier est polynomiale.

Cependant , il y a une différence importante entre les polynomes aléatoires et les polynomes sur lesquel on travaille car en général ceux-ci sont factorisables. De plus le résultat de Mignotte ne répond pas à la question de savoir si le problème de la factorisation est ou non polynomial.

5.3 Algorithme de Lenstra

Lenstra a répondu récemment à cette question (LE 82). Il donne un algorithme qui résoud le problème en un temps polynomial.

Son résultat est double. D'une part il donne un algorithme polynomial de calcul d'une base réduite dans un réseau à partir d'une base donnée , et les suites de ce résultat ne se limiterons pas uniquement à la factorisation des polynomes. Cet algorithme est assez technique et comme il sort du cadre direct de la factorisation, il ne sera pas décrit , pour cela voir l'article de Lenstra (LE 82). D'autre part il montre comment la recherche des facteurs de P peut se réaliser de manière polynomiale à l'aide de l'algorithme de base réduite.

Théorème de la base réduite dans un réseau (Lenstra , Lovacz)

Soit L un réseau de base $b_1,\ldots,b_m$, on rappelle qu'alors $L = Zb_1 + Zb_2 +\ldots+ Zb_m$, on peut construire en un temps polynomial une base B' égale à $b_0',\ldots,b_m'$ dite base réduite telle que , pour tout élément x non nul de L , $|b_0'| < 2^{m/2}|x|$. $|x|$ est la norme euclidienne de x.

Nous montrons comment la recherche d'un facteur de P devient polynomiale.

Soit $P \in Z[x]$ de degré n . Par l'algorithme de Berlekamp et la remontée de Hensel on peut calculer en un temps polynomial le polynome H_k tel que H_k divise P mod p^k

$$\deg(H_k) = l \text{ avec } l < n$$

H_k est unitaire

Supposons qu'il existe G de $Z[x]$ tel que G soit un facteur irréductible de P , H_k divise G mod p^k et $\deg(G) = m$ avec $1 \leq m \leq n-1$

D'après un résultat de Mignotte , la norme euclidienne du polynome G vérifie :

$$|G| < \binom{2m}{m}^{1/2} |P| = B$$

De plus H_k divise G entraine : $G = Q\,H_k + p^k R$ avec $\deg(R) < l$ et $\deg(Q) \leq m-l$.

Posons $Lk = Zp^k + Zp^k x +\ldots+ Zp^k x^{l-1} + ZxH_k +\ldots+ Zx^{m-l}H_k$, c'est un réseau de dimension m+1 dont une base est :

$$\begin{array}{ccccc} p^k & 0_k & 0 & & 0 \\ 0 & p^k & \cdot & & \cdot \\ \cdot & \cdot & \cdot & H_k & \cdot \\ \cdot & \cdot & \cdot^k & H_k & 0 \\ \cdot & \cdot & p^k & & \\ \cdot & \cdot & 0 & 0 & H_k \\ \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

$\leftarrow$ l éléments $\rightarrow$ $\leftarrow$ m-l+1 éléments $\rightarrow$

Le déterminant de L vaut D_k

$$\begin{vmatrix} p^k & & & & 0 \\ 0 & p^k & & & \\ & \cdot & & & \\ & & \cdot & H_k & \\ & & \cdot & p^k & \\ & & & & H_k \end{vmatrix}$$

et $D_k = p^{kl}$ car P est unitaire.

Appliquons le théorème sur les réseaux . Il existe une base réduite $b_0,\ldots,b_m$ de L_k telle que pour tout x de L_k on ait :

$$|b_0| < 2^{m/2}|x|.$$

Théorème : Si G et L_k sont définis comme ci dessus , soit V dans L_k , V = 0 et V = c G, alors :

$$p^{kl} < (|G|\,|V|)^m$$

Preuve du théorème .

V est dans L_k donc deg(V) < m.

G est irréductible et V = c G $\Rightarrow$ PGCD(G,V)=1

Donc A G + B G = 0 avec deg(A) < deg(V) entraine que A = B =0.

Par suite le réseau $L = ZG + \ldots + ZGx^{degV-1} + ZV + \ldots + ZVx^{degG-1}$ est un réseau de dimension degG + degV. Mais G et V sont dans L_k et L est un sous-réseau de L_k.

Le déterminant de L , D vérifie $D < |G|^{degV}\,|V|^{degG}$ par l'inégalité d'Hadamard. Cela donne : $D < (|G|\,|V|)^m$

L étant un sous réseau de L_k , det L > det L_k ,et donc on a le résultat:

$$p^{kl} < (|G|\,|V|)^m$$

CQFD

Le déroulement de l'algorithme est le suivant :

Choisir k tel que $p^{kl} < (2^{m/2} B^2)^m$ et construire la base réduite $b_0, b_1, \ldots, b_m$ du réseau L_k correspondant.

Alors si V est dans L_k , $|V| > 2^{m/2} B$ car $p^{kl} \leqslant (|B| \cdot |V|)^m$.

Supposons que $|G| < B$. Par construction , on a que :

$$|b_0| < 2^{m/2} |G| < 2^{m/2} B$$

La seule possibilité pour b_0 est que $b_0 = c\ G$ d'après le théorème qu'on vient de démontrer. Or b_0 est dans L_k et $b_0 = c\ G$ donc il faut que $b0 = \pm G$.

Donc on peut calculer en un temps polynomial le polynome G facteur irréductible de P de degré m , si ce facteur existe. On teste donc si ce polynome G divise P , si le test est négatif on recommence l'algorithme en cherchant un facteur irréductible de degré m+1 , jusqu'à épuisement des degrés possibles , c'est à dire m =n-1.

Il y a au plus n-1 applications de cet algorithme polynomial donc l'algorithme de factorisation de Lenstra est polynomial.

Lenstra donne la complexité suivante $O(n^{12} + n^9 (\text{Log } P)^3)$ avec les méthodes classiques de multiplication.

6 RESULTATS PRATIQUES

6.1 Méthode classique

Les résultats obtenus dans l'étude des différentes formes de remontée au chapitre 3 permettent de constater plusieurs faits.

Pour des polynomes denses , en utilisant ALDES SAC2 de COLLINS et LOOS qui effectue la factorisation modulaire pour 10 petits nombres premiers , les résultats sont les suivants :

- L'étape de factorisation modulaire prend entre 70% et 90% du

temps de calcul.

- L'étape de remontée par le lemme de Hensel prend entre 30% et 10% du temps de calcul.

- L'étape de reconstruction des facteurs dépasse rarement 1% du temps de calcul.

Ainsi l'étape en théorie la plus couteuse est en fait négligeable devant les deux autres , que ce soit la factorisation modulaire ou la remontée. On remarque que la factorisation modulaire est l'étape la plus couteuse , mais il ne faut pas oublier qu'elle est effectuée pour 10 petits nombre premiers , et que c'est l'étape de remontée qui est intrinsèquement la plus couteuse.

Si les méthodes de factorisation probabilistes permettent de faire cette factorisation modulaire avec des nombres premiers de taille moyenne , la remontée deviendra, en proportion, alors l'étape la plus importante, d'où l'interet de chercher à l'améliorer ; c'est l'objet du chapitre 3 .

6.2 Comparaisons entre méthodes

Cette partie est consacrée aux résultat d'un travail de J.Calmet et R.Loos (CA LO 82) qui ont comparé la méthode classique aux méthodes probabilistes utilisant des nombres premiers petits ou grands.

La méthode de Camion-Lazard n'était pas disponible à l'époque aussi c'est la méthode de Cantor-Zassenhauss qui a été implémentée.

Petits nombres premiers : Il n'y a pas d'interet à remplacer la méthode classique car le gain qu'on peut espérer n'est qu'une très petite part du temps de calcul.

Nombres premiers moyens : Le temps consacré à la remontée est réduit , mais le temps de calcul modulo p augmente considérablement , et en définitive la méthode ne peut pas concurrencer la méthode classique appliquée à de petits entiers premiers.

Grands nombres premiers : Il s'agit en fait de pseudo-premiers l'étape de remontée est évitée ,

mais là encore la factorisation modulaire a un cout très important que ne compense pas la disparition de la remontée.

En conclusion, il s'avère que la méthode de Berlekamp-Hensel appliquée à de petits entiers premiers est la plus interessante. Les méthodes probabilistes doivent etre améliorée pour pouvoir s'imposer, il est possible que la méthode de Camion-Lazard puisse servir de base à ces améliorations et amène à reconsidérer le problème.

6.3 Perspectives

D'un point de vue théorique le problème de la factorisation est résolu , grace à Lenstra . On peut considérer qu'il n'y a plus de découvertes importantes à faire de ce point de vue. Mais le problème théorique ne causait pas de difficultés pratiques, comme on l'a vu. Ainsi il est probable que l'algorithme de Lenstra n'apportera pas de changements car l'algorithme de calcul d'une base réduite est lent, aux dires de Lenstra lui-meme (LE 83)

Par contre l'étude de la factorisation modulo p par les algorithmes probabilistes est pleine de promesses , en particulier si on sait résoudre le problème crucial du calcul des termes en $(x^p)^1$ qui s'avère etre le plus couteux. Si ce problème était surmonté , il faudrait s'attendre à des changements spectaculaires et la factorisation se déroulerait sans doute suivant le schéma suivant :

(1) Factorisation modulo un nombre premier de taille moyenne , c'est à dire près de la taille du mot machine par des algorithmes probabilistes.

(2) Remontée des facteurs par une méthode p-adique en peu d'étape avec test de détection anticipée des facteurs.

(3) Reconstruction des facteurs restants. Cette étape serait sans doute évitée dans la plupart des cas.

Ainsi le problème théoriquement résolu de la factorisation est loin d'etre épuisé et peut etre la source de recherches interessantes.

CHAPITRE 3 : METHODES P-ADIQUES

Les méthodes p-adiques sont essentielles dans le schéma classique de la factorisation et dans de nombreux autres domaines. Après quelques rappels sur les équations diophantiennes, nous présentons les versions possibles du lemme de Hensel linéaire ou quadratique, série ou parallèle. Ensuite nous utilisons le fait que la remontée peut se présenter sous la forme de la décomposition d'une fraction rationnelle pour donner un nouvel algorithme de remontée. Les résultats de comparaisons numériques permettent d'estimer l'efficacité des algorithmes étudiés.

1 EQUATIONS DIOPHANTIENNES : SOLUTION ET REMONTEE.

1.1 Solution

Il sera nécessaire dans la suite de savoir calculer x et y solution d'équations : $xa + y\,b = c \bmod q$ où a,b,x,y,c sont des polynomes non nuls tels que a soit unitaire , q étant une puissance du nombre premier p. $\deg(y) < \deg(a)$

On fera l'hypothèse que a et b sont premiers entre eux donc il existe u et v tels que $au + bv = 1 \bmod q$ et $\deg(v) < \deg(a)$.

Proposition : Les polynomes x et y solutions de l'équation ci-dessus existent sont uniques et valent:

$$y = c\,v \bmod a\ ,\quad x = (c - y\,b)/a$$

Preuve . Unicité. S'il existe un autre couple x' et y' satisfaisant l'équation , par soustraction on obtient que :

$$(x - x')\,a + (y - y')\,b = 0$$

Comme a et b sont premiers entre eux il faut que a divise y - y', mais y - y' est de degré inférieur à celui de a , donc y - y' =0

Existence. Il suffit de montrer que les solutions données conviennent.

Par construction $\deg(y) < \deg(a)$.

$$a\,u + b\,v = 1 \bmod q \quad = \quad a\,u\,c + b\,v\,c = c \bmod q.$$

La division de v c par a est possible car a est unitaire ce qui donne : $a\,(c\,u + b\,q) + b\,y = c \bmod q$

i.e. $a\,x + b\,y = c \bmod q$

CQFD

Une autre étape nécéssaire sera de calculer les solutions x_i de $\sum_{i=1,..,n} x_i\,w_i = t$ avec $w = \prod_{j=1,..,n} r_j$ et $w_i = w / r_i$

Proposition : Les polynomes x_i solution de $\sum x_i\,w_i = t \bmod q$ $\deg(x_i) < \deg(r_i)$ où les r_i sont unitaires et premiers entre eux sont uniques et valent : $x_i = a_i\,t \bmod r_i$ où les a_i sont des polynomes tels que $\sum_{i=1 \text{ à } n} a_i\,w_i = 1 \bmod q$.

Cette proposition se démontre comme la précédente.

Remarque : Il se peut qu'un des r_i ne soit pas unitaire , la proposition reste valable si son coefficient de tete est inversible.

1.2 Remontée

Le lemme de Hensel impose de pouvoir remonter des égalités du type $a\,u + b\,v = 1 \bmod q$ en des égalités $A.U + B.V = 1 \bmod q^2$, avec A et B connus et U et V à déterminer. De plus , $A = a \bmod q$, $B = b \bmod q$, $U = u \bmod q$, $V = v \bmod q$.

On impose aussi que A est unitaire et $\deg(V) < \deg(A)$, idem pour a et v.

Proposition : Les éléments U et V cherchés sont égaux à :
$V = v + q(Tv \text{ modulo } A)$; $U = (1 - Bv)/A$
avec $T = (1 - (A.u + B.v))/q$

Démonstration. Nécessairement $U = u + qu'$ et $V = v + qv'$.
En remplaçant U et V par leur valeur dans l'égalité voulue il arrive : $A.u + B.v = (1 - (A.u + B.v))/q$ ce qui donne le résultat cherché d'après la partie précédente sur les équations diophantiennes.

De meme il faut remonter des égalités $\sum a_i.w_i = 1$ en des égalités $\sum A_i.W_i = 1 \bmod q^2$ avec $w_i = w/r_i$ et $w = \prod r_i$; idem pour W_i, et $R_i = r_i \bmod q$. On suppose toujours que les r_i sont unitaires et premiers entre eux , donc les R_i aussi.

Proposition : les A_i correspondant au problème ci-dessus valent:
$A_i = a_i + q(Ta_i \bmod r_i)$ avec $T = (1 - \prod a_i.w_i)/q$

La démonstration est semblable à la précédente.

2 LEMME DE HENSEL

2.1. Forme linéaire et quadratique.

2.1.a Forme linéaire.

Il s'agit de la forme exposée au chapitre 2 , donc il s'agit ici d'un rappel succint.

Proposition : Soit P un polynome primitif , sans carré , de degré n , soit p un nombre premier , soit a un facteur unitaire de P modulo p et b = P/a. Alors pour tout entier k , il existe a_k et b_k tels que : $P = a_k . b_k \bmod p^k$

$$a_k = a \bmod p \quad \text{et} \quad b_k = b \bmod p$$

La démonstration et le calcul effectif de a_k et b_k se trouvent au chapitre 2 , en 2.3.1

L'algorithme qui permet d'obtenir tous les facteurs modulo p^k d'un polynome P est le suivant :

Entrée : P polynome primitif sans carré , $L = r_1,\ldots,r_r$ liste des facteurs de P modulo p un nombre premier bien choisi.

Sortie : Ls liste des facteurs de P modulo p^k.

(1) $P_1 = P$, $a_1 = r_1$, $b = P/a_1$, $L = L - \{r_1\}$
(2) Tant que card(L) > 1 Faire

début
(2.1) Calculer a_k et b_k remontée modulo p^k des facteurs a_1 et b_1 du polynome P_1, par le lemme de Hensel.
(2.2) $Ls = Ls \cup \{a_k\}$, $P_1 = b_k$
fin

(3) $Ls = Ls \cup \{P_1\}$
fin

L'idée est donc de remonter le facteur a de P en ak , puis de remplacer P par P/ak et de recommencer ,ainsi chaque fois l'algorithme de remontée s'applique à des polynomes dont le degré décroit , ce qui va permettre d'obtenir une complexité comparable à celle des algorithmes parallèles.

2.1.b Forme quadratique

La méthode précédente a l'inconvénient de faire augmenter que faiblement le module à chaque itération puisqu'il passe de pk à pk+1.Or on a vu que les bornes qui interviennent dans la factorisation sont élevées et que les nombres premiers utilisés sont petits. Il est interessant de remonter les facteurs en réalisant moins d'itérations , c'est ce que réalisent les méthodes quadratiques. Celles-ci remontent à la fois l'égalité P = ab et l'identité au + bv = 1.

Proposition : Soit P un polynome primitif, sans carré, a et b facteurs de P modulo le nombre premier p , a étant unitaire , alors pour tout k il existe a_k et b_k , u_k et v_k tels que :

$$P = a_k.b_k \bmod q$$

$$a_k = a \bmod p \text{ et } b_k = b \bmod q \quad \text{avec } q = p^{2^k}$$
$$a_k.u_k + b_k.v_k = 1 \bmod q$$

L'algorithme qui correspond au calcul de a_k et b_k est le suivant :

Entrée : P polynome primitif sans carré , a un facteur unitaire de P , b = P/a , modulo p.

Sortie : a_k et b_k facteurs modulo p^{2^k} de P.

(1) Calculer u et v tels que $au + bv = 1 \bmod p$
$\deg(v) < \deg(a)$

(2) $a_1 = a$, $b_1 = b$, $u_1 = u$, $v_1 = v$, $q = p$, $i = 1$

(3) Tant que $i < k$ Faire

début

(3.1) $s = (P - a_i.b_i)/q$

(3.2) $as_i = v_i.s \bmod(a_i,q)$

(3.3) $a_{i+1} = a_i + qas_i$, $b_{i+1} = P/a_{i+1}$

(3.4) $t = (1 - (ui.ai+1 + vi.bi+1))/q$

(3.5) $vs_i = tv_i \bmod(a_i,q)$

(3.6)

$$v_{i+1} = v_i + qvs_i \text{ , } u_{i+1} = (1 - b_{i+1}.v_{i+1})/a_{i+1}$$

(3.7) $i = i+1$

fin

(4) $L = \{a_k, b_k\}$

fin

Les étapes 3.1 à 3.3 correspondent à la remontée de $P = a_i.b_i \bmod q$ en $P = a_{i+1}.b_{i+1} \bmod q^2$; les étapes 3.4 à 3.7 à la remontée de $a_iu_i + b_iv_i = 1 \bmod q$ en $a_{i+1}u_{i+1} + b_{i+1}v_{i+1} = 1 \bmod q^2$.

Preuve de la proposition et de l'algorithme.

Montons qu'à l'itération i , $P = a_i.b_i \bmod q$ avec $q = p^{2^i}$ et que $a_iu_i + b_iv_i = 1 \bmod q$ avec $\deg(v_i) < \deg(a_i)$ et $a_i = a \bmod p$.

Cela est vrai à l'itération 1.

Supposons le vrai à l'itération i.

Si $P = A_{i+1} \cdot B_{i+1} \bmod q^2$ alors nécessairement $A_{i+1} = a_i + qa_i'$, $B_{i+1} = b_i + qb_i'$

donc $P = a_i b_i + q(a_i' b_i + a_i b_i') + q^2 a_i' b_i' \bmod q^2$

d'où $(P - a_i b_i)/q = a_i b_i' + b_i a_i' \bmod q$

i.e. $s = a_i b_i' + b_i a_i' \bmod q$ et d'après les résultats sur les équations diophantiennes, on a : $a_i' = as_i$ et par suite $A_{i+1} = a_i + qas_i = a_{i+1}$ et par conséquent $B_{i+1} = b_{i+1}$.

De plus par construction $a_{i+1} = a \bmod p$ et $\deg(a_{i+1}) = \deg(a)$.

La validité des résultats pour v_{i+1} et u_{i+1} se montre de manière identique en utilisant les résultats vus sur les équations diophantiennes.

CQFD

2.1.c Comparaison et étude de complexité

Comparaison

L'algorithme linéaire remonte l'égalité $P = ab \bmod q$; l'algorithme quadratique remonte les deux égalités $P = ab \bmod q$ et $au + bv = 1 \bmod q$.

Les calculs de remontée sont du meme type pour chacune des équations considérées car il s'agit à chaque fois de résoudre une équation diophantienne du meme type ,les polynomes qui interviennent ayant un degré majoré par le degré de P.

Donc si on reste dans la limite du mot machine , la taille des coefficients ne joue pas et une remontée quadratique coute le double d'une remontée linéaire. Mais d'un coté on effectue k itérations , et de l'autre seulement $\log_2(k)$ itérations. Donc la remontée quadratique apparait préférable.

Si la taille du mot machine est dépassée il n'est plus possible de conclure théoriquement du moins car les complexités sont alors équivalentes.

Seuls les algorithmes quadratiques seront exposés, d'autant plus que les nombres premiers utilisés dans la factorisation sont petits et que les bornes rencontrées sont grandes , donc la différence entre k et $\log_2(k)$ est importante.

Complexité

Soit $M_q(r,s)$ le cout de la multiplication de deux polynomes de degrés respectifs r et s dont les coefficients sont majorés par q. On a $M_q(r,s) = k(r,s)Log^2(q)$

Soit n le degré de P et s le degré de a_i , D un majorant de la valeur absolue des coefficients de P.

Le cout de la boucle 3 est :

étape 3.1 : $M_q(n,s)$

étape 3.2 : $O(M_q(n,s))$

étape 3.3 : $O(M_q(n,s))$

De meme les étapes 3.4 à 3.6 donnent un cout en $O(M_q(n,s))$.

Montrons qu'une étape coute aussi cher que toutes les précédentes.

Les étapes 1 à i-1 coutent: $(1 + 2^2 + \ldots + 2^{2(i-1)})k(n,s)Log^2(p)$

Montrons par réccurence que : $1 + 2^2 + \ldots + 2^{2(i-1)} < 2^{2i}$.

C'est vrai pour i = 1 , supposons le vrai pour i-1.

Alors $(1 + 2^2 + \ldots + 2^{2i}) = 1 + \ldots + 2^{2(i-1)} + 2^{2(i-1)}$

$< 2^{2(i-1)} + 2^{2(i-1)} < 2^{2i}$

Donc une étape de remontée quadratique coute aussi cher que toutes les précédentes , en particulier la dernière étape coute aussi cher que toutes les autres ; donc le cout de remontée d'un facteur est celui de cette étape et vaut : $O(M_M(n,s))$

Pour diminuer ce cout , à la dernière étape , au lieu de travailler modulo une puissance qudratique de p , on travaille modulo la borne M.

exemple : Pour p = 11 et $M = 11^6$, le module de la dernière étape serait de $q = 11^8 = 121$ M. On travaillera modulo M on lieu de q = 121 M à la dernière itération.

Calculons le cout de remontée de tous les facteurs.

Pour cela on suppose que les r facteurs de P ont tous le meme degré moyen n/r . La remontée de tous ces facteurs coute :

$$O(M_q(n,n/r) + M_q(n(r-1)/r,n/r) + \ldots + M_q(2n/r,n/r))$$

avec q = M , d'ou la proposition suivante.

Proposition : Le cout de la remontée de Hensel par la méthode série quadratique est $O(n^4 \text{Log}^2(D))$ si on utilise les méthodes classiques.

Si on utilise la FFT , $Mq(r,s)$ vaut $t\text{Log}(t)$ si t est le maximum de r et s. Il faut évaluer S qui vaut :

$$S = n\text{Log}n + n(r-1)/r\text{Log}(n(r-1)/r) + \ldots + 2n/r\text{Log}(2n/r)$$

Il s'agit d'une somme de Riemann de la fonction croissante positive $x \to x\text{Log}(x)$ et une très bonne approximation de S est l'intégrale de cette fonction entre n et n/r quotientée par r/n. D'où $S = O(rn\ \text{Log}(n))$

Proposition : Le cout de la remontée série quadratique de tous les facteurs est $O(rn^3\ \text{Log}(n)\ \text{Log}^2(D))$ pour la FFT.

2.2 Forme paralléle

2.2.1 Description

Il s'agit d'une version du lemme de Hensel qui permet de remonter simultanément tous les facteurs. Les algorithmes sont dus à Wang (WA 79).

Algorithme de remontée simultanée

Entrée : P polynome primitif sans carré, $r_1,\ldots,r_r$ ses facteurs unitaires modulo p

Sortie : $L = r_{k1},\ldots,r_{kr}$ les facteurs unitaires de P modulo p^{2^k}

(1) (Initialisation) i=1 , q=p ,Pour j=1 à r , $r_{1j}=r_j$

(1.1) $r_{11}=a_n r_{11}$ avec a_n coefficient de tete de P .

(1.2) $w = \prod r_{ji}$

(1.3) Pour j=1 à r , $w_j = w/r_j$

(1.4) Calculer a_i tels que $\sum_{i=1 \text{ à } r} a_i w_i = 1 \bmod p$
$\deg(a_i) \quad \deg(r_i)$

(2) Tant que $i < k$ Faire

début

(2.1) $as = (P - \prod r_{ij})/q$, $r_{i1} = a_n^{-1} r_{i1} \bmod q$

(2.2) Pour j=1 à r Faire
(remontée des facteurs)

début

(2.2.1) $r_{i+1,j} = r_{ij} + q(a_j \text{ as mod } r_{ij})$, $r_{i+1,1} = a_n r_{i+1,1}$

fin

(2.3) i=i+1 , si i=k retourner L $=\{r_{i1},\ldots r_{ir}\}$

(2.4) $w = \prod r_{ij}$, $w_j = w/r_{ij}$

(2.5) (remonter l'égalité $\sum a_i w_i = 1 \bmod q$ en $\sum a_i w_i = 1 \bmod q^2$)

$t = (1 - \sum a_j w_j)/q$

Pour j=1 à r Faire

début

(2.5.1) $a_j = a_j + q(a_j t \bmod r_{ij})$

fin

(2.6) $q = q^2$

fin

fin

Remarques - Toutes les divisions sont effectuées avec un diviseur unitaire.

- Les étapes 2.1 à2.2.1 remontent les facteurs , les étapes 2.5 et 2.5.1 remontent les coefficients a_i de l'identité de Bezout $\sum a_i w_i = 1$.

- Les multiplications par a_n ou a_n^{-1} du premier facteur sont nécessaires pour que $(P - \prod^n r_{ij})^n$ ait un degré inférieur à celui de P.

Preuve de l'algorithme : Il suffit de montrer qu'à l'étape i :

$$P = \prod r_{ij} \bmod q \text{ et } \sum a_i w_i = 1 \bmod q$$

C'est vrai à l'étape 1.

Supposons le vrai à l'étape i.

Si $P = R_1 \ldots R_r \bmod q^2$ alors $R_j = r_{ij} + qr'_j$.

Comme le coefficient de tete de r_{i1} est égal à celui de P , alors $\deg(r'_j) < \deg(r_{ij})$ et l'égalité précédente donne :

$$(P - \prod r_{ij})/q = \sum r'_j w_j \bmod q \text{ avec } \deg(r'_j) < \deg(r_{ij})$$

D'après les résultats sur les équations diophantiennes vus a début du chapitre, les r'_j correspondent à ce qui est calculé en 2.2.1 donc $R_j = r_{i+1,j}$.

De meme les étapes suivantes remontent bien l'identité de Bezout sur les a_i et w_i (meme démonstration).

CQFD

2.2.2 Algorithme d'initialisation

Il s'agit de l'algorithme calculant les polynomes a_i tels que $\sum_{i=1 \text{ à } r} a_i w_i = 1 \bmod p$ avec $\deg(a_i) < \deg(r_i)$

Les calculs se font dans $Z_p[x]$, donc tous les coefficients non nuls sont inversibles et bornés par p ; donc l'algorithme d'Euclide peut etre utilisé , de plus l'explosion de la taille des coefficients qui se produit sur les entiers est évitée. Les r_i sont des polynomes premiers entre eux 2 à 2 donc les a_i existent.

Algorithme d'initialisation.

Entrée : $r_1,\ldots,r_r$ polynomes premiers 2 à 2.

Sortie : L liste des polynomes a_i tels que $\sum a_i w_i = 1 \bmod p$ et $\deg(a_i) < \deg(r_i)$

(1) Calculer par l'algorithme d'Euclide a_1 et b_1 tels que :

$a_1 w_1 + b_1 r_1 = 1 \bmod p$ et $\deg(a_1) < \deg(r_1)$

(2) i = 2 , Tant que i < r-1 Faire

début

(2.1) Calculer as_i et bs_i par l'algorithme d'Euclide , tels que $as_i(r_{i+1}\ldots r_r) + bs_i\, r_i = 1$ et $\deg(as_i) < \deg(r_i)$

(2.2) $c_i = b_{i-1}$

(2.3) Calculer a_i et b_i tels que $a_i(r_{i+1}\ldots r_r) + b_i r_i = c_i$ $\deg(a_i) < \deg(r_i)$

(2.4) i=i+1

fin

(3) $a_r = b_{r-1}$, $L = \{a_1,\ldots,a_r\}$

fin

Preuve : Il suffit de montrer qu'à l'étape i :

$\sum_{j<i} a_j w_j + b_i(r_1\ldots r_{i-1}) = 1$ avec $\deg(a_i) < \deg(r_i)$

C'est vrai à l'initialisation. Supposons le vrai à l'étape i. On sait que a_{i+1} et b_{i+1} vérifient :

$a_{i+1}(r_{i+2}\ldots r_r) + b_{i+1}r_{i+1} = b_i$ donc en remplaçant b_i par cette expression dans l'égalité précédente on obtient :

$$\sum_{j<i+1} a_j w_j + b_{i+1}(r_1\ldots r_i) = 1$$

A la dernière étape $\sum a_j w_j + b_{r-1}w_r = 1$ donc comme $a_r = b_{r-1}$, le résultat est démontré.

CQFD

Une autre méthode est suggérée par Waezi (WA 83) :

$$\left[\begin{array}{l}\sum_{i=1 \text{ à } r} a_i\, w_i = 1 \\ \deg(a_i) < \deg(r_i)\end{array}\right] \Leftrightarrow \left[\begin{array}{l}\sum_{i=1 \text{ à } r} a_i/r_i = 1/w \\ \deg(a_i) < \deg(r_i)\end{array}\right]$$

Cela définit les a_i comme solution de la décomposition en éléments simples de la fraction rationnelle $1/w$, on utilise alors l'algorithme de Kung et Tong (KU TO 77).

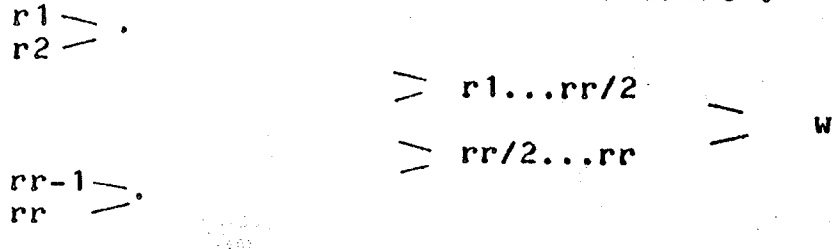
2.2.3 Etude de complexité de l'algorithme parallèle

a Méthode classique : la partie couteuse est le calcul de w qui est calculé comme suit :
$w = (r_1(r_2(r_3(\ldots r_r))))$ et donc le cout de ce calcul est de :

$$(n/r.n/r + n/r.2n/r + \ldots + n/r.(r-1)n/r) \log^2(q)$$

ce qui donne un cout total en $O(n^4 \text{Log}^2(D))$ pour les méthodes classiques.

b Méthode arborescente : Le calcul de w s'effectue à l'aide d'un arbre :



Alors le calcul de w et de tous les produits intermédiaires s'effectue en M(n)Log(r) opérations. On peut montrer que tous les calculs de l'algorithme peuvent , en utilisant les résultats intermédiaires, s'effectuer en O(M(n) Log(r)) opérations . Une étude détaillée pour les fractions rationnelles se trouve dans l'article de Kung et Tong (KU TO 77), et les méthodes utilisées

s'appliquent immédiatement pour le cas de l'algorithme étudié ici.

L'utilisation d'arbre n'apporte pas d'améliorations si on utilise les méthodes classiques de multiplication , cela est du à une majoration trop forte de chaque étage de l'arbre.

c Méthode FFT et arborescente : Tous les calculs de l'algorithme peuvent se faire en $O(M(n)Log(r))$ à l'aide d'arbres et en utilisant les résultats auxiliaires trouvés. Les techniques utilisées se trouvent dans l'article précité de Kung et Tong . On a alors le résultat que le cout de l'algorithme de remontée est :

$O(n^3 Log(n) Log(r) Log^2(d))$ en utilisant la FFT et les arbres.

Le cout de l'algorithme de remontée avec la FFT ,sans utilisation d'arbre est :

$O(rn^3 Log(n) Log^2(D))$.

Il y a une amélioration par rapport à la méthode série si on utilise la technique des arbres ,puisqu'un facteur r devient un facteur Log(r). En fait r étant le nombre de facteurs , cette amélioration est négligeable et ne compense pas le surcout occasionné par la place mémoire supplémentaire utilisée et la complexité accrue de l'algorithme. Seule les comparaisons pratiques peuvent départager les deux méthodes , série ou parallèle.

3 DECOMPOSITION EN ELEMENTS SIMPLES ET LEMME DE HENSEL

3.1 Algorithme de Kung et Tong.

C'est l'algorithme le plus performant connu pour la décomposition en éléments simples d'une fraction rationnelle sur $K[x]$, où K est un corps.

Algorithme de Kung et Tong

Entrée : $P/R_1 \ldots R_r$ une fraction rationnelle irréductible; les R_i sont des polynomes premiers entre eux , $\deg(P) < \sum \deg(R_i)$

Sortie : Liste $L = C_1, \ldots, C_r$ avec $P/R_1 \ldots R_r = \sum C_i/R_i$ et $\deg(C_i) < \deg(R_i)$

(1) $W = \prod R_i$

(2) Pour i = 1 à r Faire

début

(2.1) $W_i = W/R_i$

(2.2) $D_i = W_i \bmod R_i$

(2.3) Calculer par l'algorithme d'Euclide A_i et E_i tels que $A_i.D_i + E_i.R_i = 1$ et $deg(A_i) < deg(R_i)$

(2.4) Si P = 1 alors $C_i = A_i$

(2.5) $L_i = P \bmod R_i$

(2.6) $C_i = A_i.L_i \bmod R_i$

fin

(3) $L = \{C_1, \ldots, C_r\}$

fin

La preuve de l'algorithme se trouve dans Kung et Tong (KU TO 77) . Elle est d'abord donnée quand P = 1 et repose sur le fait que les C_i et les solutions X_i de $P/R_1 \ldots R_r = \Sigma X_i/R_i$ sont solution du meme problème d'interpolation d'Hermitte. Il est ensuite facile d'étendre la solution au cas P quelconque.

Remarque : L'algorithme suppose que les polynomes sont définis sur K[x] où K est un corps , mais s'il est possible de diviser par R_i et de calculer A_i et E_i tels que $A_i.D_i + E_i.R_i = 1$ dans A x où A est un anneau , il reste valable. En particulier pour $A = Z_q$ et si les R_i ont un coefficient de tete inversible , on peut l'utiliser.

La complexité de cet algorithme est O(Log(r)M(n)).

3.2 Lien avec le lemme de Hensel

3.2.1. Présentation

A l'étape k , la remontée des facteurs s'écrit:
Trouver r_i' tels que $P = \Pi (r_i + q\, r_i') \bmod q^2$

$$deg(r_i') < deg(r_i)$$

C'est à dire tels que $S / r_1 \ldots r_r = \Sigma r_i' / r_i \bmod q$

$$deg(r_i') < deg(r_i)$$
avec $S = (P - \Pi r_i)/q$

Donc les r_i' sont solutions de la décomposition en éléments simples de la fraction $S / r_1 \ldots r_r$ dans $Z_q[x]$.

Il est possible d'utiliser l'algorithme de Kung et Tong pour réaliser la remontée , c'est ce que fait le nouvel algorithme proposé. Cette idée est due à Viry (VI 81), mais il ne propose pas d'algorithme dans un anneau et impose que les polynomes soient unitaires.

Algorithme de remontée par décomposition en éléments simples.

Entrée : P polynome primitif sans carré, $r_1,\ldots,r_r$ ses facteurs unitaires modulo p

Sortie : Liste $L = r_{k1},\ldots,r_{kr}$ des facteurs unitaires de P modulo p^{2^k}

(1) (initialisation) i=1 , q=p , Pour j=1 à r , $r_{1j}=r_j$

(1.1) $w = a_n \prod r_{ij}$ avec a_n coefficient de tete de P

(1.2) Pour j=2 à r Faire

début
(1.2.1) $w_j = w/r_{ij}$, $d_j = w_j \bmod r_{ij}$

(1.2.2) Calculer a_j et e_j tels que $a_j d_j + e_j r_{ij} = 1 \bmod p$ $\deg(a_j) < \deg(r_{ij})$

fin

(2) Tant que i < k Faire

début
(2.1) Si i=1 alors as = $(P - \prod r_{ij})/q$ sinon as = ak/q

(2.2) Pour j = 2 à r Faire

début
(2.2.1) (remontée des facteurs autres que le premier)
$r_{i+1,j} = r_{ij} + q(a_j as \bmod r_{ij})$
fin

(2.3) $P = (\prod_{j>1} r_{i+1,j}).q_1 + ak \bmod q^4$, $r_{i+1,1}=q_1$

(2.4) i=i+1 , si i = k alors $r_{i+1,1} = a_n^{-1} r_{i+1,1} \bmod q^2$ retourner la liste L des facteurs $a_n r_{ij}$

(2.5) w = P mod q2

(2.6) Pour j = 2 à r Faire

début
(2.6.1) $w_j = w/r_{ij}$, $d_j = w_j \bmod r_{ij}$

(2.6.2) Remonter $a_j d_j + e_j r_{ij} = 1 \bmod q$ en $a_j d_j + e_j r_{ij} = 1 \bmod q^2$

fin

(2.7) $q = q^2$

fin

fin

Les coefficients ai et ei qui interviennent sont les memes que ceux notés Ai et Ei dans l'algorithme de Kung et Tong. Ils sont ceux relatifs à la décomposition en éléments simples de la fraction $((P - \prod rij)/q)/ri1 \ldots rir$.

Les différences essentielles avec le lemme de Hensel parallèle sont les suivantes :

- Le premier facteur est calculé par division de P par les autres facteurs et le reste de cette division est égal à P moins le produit des facteurs.

- Pour chaque facteur , on calcule les coefficients ai et ei indépendament des autres facteurs, ce qui assure l'indépendance des calculs relatif à chaque facteur.

3.2.2 Preuve de l'algorithme.

Démontrons d'abord que les coefficients a_j et e_j sont ceux relatifs à la décomposition en éléments simples de la fraction $ak/r_{i1} \ldots r_{ik}$.

Par constuction ils vérifient $a_j d_j + e_j r_{ij} = 1 \bmod q$ avec $d_j = w_j$ et $w_j = \prod r_{il} / r_{ij}$; d'où le résultat.

Le terme ak est égal par construction à $P - \prod r_{ij} \bmod q^2$, donc pour $j > 2$, les termes $a_j a_s \bmod r_{ij}$ sont les solutions de la décomposition en éléments simples de $as / r_{ij} \ldots r_{ir}$. Il en résulte que les $r_{j+1,j}$ sont les facteurs unitaires de P modulo q^2 pour $j > 2$ et le terme $r_{i+1,1}$ est par construction le premier facteur unitaire de P modulo q_2 au coefficient de tete près.

CQFD

3.2.3 Complexité

Les calculs effectués par le nouvel algorithme utilisent les memes quantités que l'algorithme du lemme de Hensel , donc les calculs les plus couteux sont les memes et les complexités vont etre identiques.

La complexité du nouvel algorithme est :

O(n4Log2(D)) avec les méthodes classiques

O(rn3Log(n)Log2(D)) avec FFT

O(n3Log(n)Log(r)Log2(D)) avec FFT et utilisation d'arbres.

3.2.4 Forme série du nouvel algorithme

Quand le nombre de facteurs est égal à 2 , les algorithmes paralléles redonnent des formes proches des formes série. On peut proposer une forme série du nouvel algorithme qui est proche de la forme série du lemme de Hensel. La différence esentielle est qu'on calcule un des facteur par division et que le reste de cette division estutilisé pour calculer l'approximation suivante.

Algorithme de remontée d'un facteur.

Entrée : P polynome primitif sans carré, a un facteur unitaire de P modulo p.

Sortie : a_k facteur unitaire de P modulo p^{2^k} et b_k son cofacteur

(1) (initialisation) $P = aq_1 + r \bmod p^2$ avec $deg(r) < deg(a)$
$q=p$, $a_1 = a$, $b_1 = q_1$, $r = r/p$
Calculer u et v tels que $au + bv = 1 \bmod q$
$u_1 = u$, $v_1 = v$

(2) Tant que $i < k$ Faire

début

(2.1) Calculer as_1 solution de $as_1 b_i + a_i bs_1 = r$ à l'aide de $a_i u_i + b_i v_i = 1 \bmod q$

(2.2) $a_i = a_i + qas_1$

(2.3) $P = a_i . q_i + r \bmod q^4$ avec $deg(r) < deg(a_i)$

(2.4) $b_i = q_i$, $r = r/q$

(2.5) Remonter $a_i u_i + b_i v_i = 1 \bmod q$ en $a_i u_{i+1} + b_i v_{i+1} = 1 \bmod q^2$

(2.6) $u_i = u_{i+1}$, $v_i = v_{i+1}$, $q = q^2$, $i=i+1$

fin

fin

La validité de cet algorithme se montre comme celle de l'algorithme parallèle correspondant dont il n'est que la traduction sous forme série.

La remontée de tous les facteurs se fait comme celle vue lors de l'étude du lemme de Hensel série.

4 PROBLEMES ET AMELIORATIONS

4.1 Detection anticipée des facteurs et réinitialisation

On a vu au chapitre 2 ,&4.2 qu'il était intéressant de chercher à trouver les facteurs de P avant d'atteindre la borne M et l'étude de complexité qui montre qu'une étape est aussi couteuse que toutes les précédentes ,incite encore plus à effectuer la détection anticipée des facteurs. Donc dès que le module courant q dépasse une borne heuristique m", on teste si les facteurs modulaires obtenus donnent de vrais facteurs de P.

Si cela est le cas on calcule la borne correspondant au nouveau polynome à factoriser et si elle est dépassée par le module atteint ,on passe à l'étape de reconstruction des facteurs. Mais si elle est supérieure au module courant , il faut reinitialiserles algorithmes parallèles .

Réinitialisation du nouvel algorithme.

On suppose que $P = r_1 \ldots r_r$ modulo q et que certains des r_i donnent des facteurs de P sur Z. On peut supposer sans se restreindre que pour i $\leq$ l ,les r_i donnent des facteurs de P . La nouvelle borne est supposée supérieure au module q ; il faut donc calculer les nouveaux coefficients a_i et e_i.

On note en capitales les nouvelles indéterminées ,soit A_i,E_i, R_i, D_i, W_i.

On a que $R_i = r_i$ et que $W_i = w_i/F$ avec $F = r_{l+1} \ldots r_r$

Proposition : Les coefficients A_i réinitialisés sont solution de $A_i d_i + G_i r_i = F \bmod q$ donc $A_i = Fa_i \bmod r_i$

Preuve : On cherche A_i et E_i tels que $A_i D_i + E_i D_i = 1 \bmod q$ et $\deg(A_i) < \deg(R_i)$

On connait:

a_i et e_i tels que $a_i d_i + e_i r_i = 1 \bmod q$ et de plus on sait que $D_i = W_i \bmod R_i$ donc que $FD_i = d_i \bmod r_i$ car $R_i = r_i$.

Par suite $A_i(FD_i) + E_iFR_i = F$ et d'après les résultats sur les équations diophantiennes déjà vus $A_i = Fa_i \bmod r_i$

CQFD

Les coefficients ai du nouvel algorithme sont en fait les memes que ceux du lemme de Hensel parallèle puisqu'ils vérifient $\sum aiwi = 1 \bmod q$; donc la réinitialisation proposée , naturelle avec l'approche des fractions rationnelles , est valable aussi pour le lemme de Hensel parallèle .

4.2 Lien avec la méthode de Newton

Quand un facteur du polynome est linéaire et unitaire , il est possible d'améliorer le nouvel algorithme. Soit $r_i = X - x$ ce facteur.

Etant donnée une appproximation x_i de ce facteur modulo q , il faut calculer plusieurs polynomes modulo r_i pour obtenir l'approximation modulo q^2. Mais diviser par r_i revient à prendre la valeur en x_i des polynomes considérés.

En particulier , le calcul de d_i peut se faire de la façon suivante :

$d_i = (DP(x)/Dx)(x_i)$ où $DP(x)/Dx$ est la dérivée du polynome P par rapport à x.

Si d_i est un élément de Zq , les a_i et e_i correspondants valent $e_i = 0$ et $a_i = 1/d_i \bmod q$.

Donc l'approximation modulo q^2 du facteur vaut:

$$X - (x_i - q((P(x_i)/q)/P'(x_i)) \quad \text{si } P'(x_i) = (DP(x)/Dx)(x_i)$$

et l'approximation modulo q^2 de la racine x est :

$$x_{i+1} = x_i - q((P(x_i)/q)/P'(x_i))$$

Il s'agit d'une formule analogue à la formule de Newton pour le calcul des approximations successives d'un zéro de P. On dispose donc d'une méthode rapide de remontée des facteurs unitaires de degré 1 qui nécessite une dérivation et deux évaluations. Cela donne une méthode de calcul des zéros d'un polynome à coefficients entiers , ce qui est interessant dans de nombreux cas. L'article de Loos (LO 83) fait le point sur ce problème en particulier sur le lien avec la méthode de Newton.

5 COMPARAISONS ET MESURES

5.1 Préliminaires

Nous avons comparé les différents algorithmes de remontée en utilisant le système de calculformel ALDES/SAC2 de Collins et Loos, implementé sur un CII-HB DPS 68 fonctionnant sous MULTICS. Dans ce système la factorisation modulo p réalise d'abord la factorisation en produit d'irréductibles de meme degré puis ceux-ci sont factorisés par l'algorithme de Berlekamp. La factorisation modulaire est effectuée pour 10 petits nombres premiers et on retient le plus petit nombre premier qui donne un nombre minimal de facteurs. Cette méthode permet de réduire le nombre de facteurs parasites.

Les polynomes utilisés étaient denses et leurs coefficients avaient en général le meme nombre de chiffre. La taille du mot machine dans l'implémentation réalisée était de $2^{32} - 1$ soit un nombre d'environ 10 chiffres.

Notations.

- Le polynome à factoriser est appelé a, son degré est deg(a).

- T est le temps requis pour la factorisation modulaire (10 petits nombres premiers utilisés)

- ti est le temps nécessité par la remontée des facteurs.

  i = 1 : remontée par le lemme de Hensel série.

  i = 2 : remontée par le lemme de Hensel série avec la détection des facteurs.

  i = 3 : remontée par le lemme de Hensel parallèle avec détection des facteurs.

  i = 4 : remontée par le nouvel algorihme basé sur les fractions rationnelles avec détection des facteurs.

  i = 5 : remontée par la forme série du nouvel algorithme avec détection des facteurs.

- d est le nombre moyen de chiffres du polynome a , on indique "mot machine" si d est de l'ordre du mot machine.

Tous les temps sont donnés en millisecondes.

5.2 Résultats

Le temps nécessité pour la remontée dépend du nombre de facteurs trouvés modulo p , et surtout de l'existence ou non de facteurs parasites. Le cas le plus mauvais ne s'est pas présenté sauf avec des polynomes du type $x^4 + 1$ irréductible sur Z[x] et réductible modulo p pour tout nombre premier p. Ce cas n'est pas très significatif , nous donnons les temps obtenus pour $x^4 + 1$.

T = 2411ms , t1 = 881ms , t2 = 902ms , t3= 1080ms ,
t4 = 892ms , t5 = 689ms.

Il faut remarquer que ce cas est le pire pour les algorithmes parallèles puisqu'aucune détection de facteurs n'a lieu.

Nous donnons maintenant les résultats plus généraux relevés.

deg(a) = 8

d	4	4	4	8
T	17897	17310	17501	17706
t1	3656	3395	5311	5874
t2	2116	1902	3027	4174
t3	3787	2661	5186	8198
t4	1910	1892	3472	5353
t5	1937	1899	2951	3708

Le mauvais résultat (8198ms) obtenu pour le dernier polynome par le lemme de Hensel parallèle , s'explique par la détection anticipée d'un facteur . Le module courant étant insuffisant il a fallu réinitialiser l'algorithme . La réinitialisation choisie pour cet algorithme est le redemarrage de la remontée pour les facteurs restants au tout début , d'où le cout élevé. Par contre la réinitialisation du nouvel algorithme parallèle qui est celle décrite en 4.1 ne le pénalise pas trop (5353ms).

deg(a) = 8

d	mot machine	mot machine
T	19241	18424
t1	3875	3997
t2	3814	3128
t3	5619	4631
t4	3328	2636
t5	4897	2316

deg(a) = 12

d	5	5	9	mot machine
T	51917	47441	56341	47510
t1	9828	17652	15341	12741
t2	5374	11069	8048	7965
t3	7491	16371	8844	10910
t4	5714	11099	6878	8969
t5	4676	6677	5830	6495

deg(a) = 20

d	5	5	5	mot machine
T	206522	187395	181411	195086
t1	31513	34771	30831	28749
t2	14997	29486	15912	10739
t3	25247	32001	27170	17802
t4	17907	22289	20144	10809
t5	13495	17829	13490	8014

5.3 Commentaires

Plusieurs faits apparaissent :

- Les algorithmes série se revèlent etre les meilleurs dès qu'on ajoute la possibilité de détection anticipée des facteurs.

- Le nouvel algorithme sous sa forme série est le plus rapide , sa forme parallèle vient en seconde position à égalité avec le lemme de Hensel série.

- La détection anticipée des facteurs joue un role essentiel dans la réduction du temps de remontée , à condition que la factorisation modulaire ne soit pas trop mauvaise.

- Le temps de reconstruction des facteurs a été incorporé dans le temps de remontée pour ne pas pénaliser la détection anticipée , mais il est apparu négligeable en pratique (moins de 1% du temps total)

- Il est intéressant de noter la progression très rapide du temps de la factorisation modulaire en fonction du degré ; ainsi que la progression du temps de remontée en fonction de la taille des coefficients. Une division par 10 (nombre de factorisations modulaires effectuées) du temps T permet de mieux cerner l'importance de l'étape de remontée qui devient prépondérante. Les améliorations proposées prennent alors toutes leur importance et permettent de réduire de façon appréciable le temps de la factorisation puisque dans le meilleur des cas le temps de remontée est divisé par 3.

6 CONCLUSION

Au terme de cette étude des méthodes p-adiques utilisées dans la factorisation des polynomes à coefficients entiers, plusieurs résultats s'imposent.

La détection anticipée des facteurs joue un role capital , et les algorithmes "série" sont les plus performants , contrairement à l'opinion généralement répandue sur la question.

La nouvelle méthode , série ou parallèle , est la plus efficace .Sa forme parallèle est efficace , et le problème du redémarrage après détection de facteurs est parfaitement résolu.

D'un point de vue théorique , la complexité des méthodes parallèles est meilleure , à condition d'utliser des arbres pour presque tous les calculs et uniquement dans le cas d'utilisation des méthodes de multiplication rapide (FFT) . En effet on obtient un facteur Log(r) au lieu d'un facteur r ; cependant ce résultat n'est pas interessant en pratique.

Le problème du choix d'un nombre premier optimal et du lien entre le choix de ce nombre premier et la factorisation du polynome reste posé et semble très difficile à résoudre.

Enfin la factorisation des polynomes creux n'a pas été abordée , pour cela on peut se reporter à la thèse de Zippel (Zi 79) qui propose une forme adaptée du lemme de Hensel et à l'article de Davenport (DA 83) qui traite de certains cas particuliers.

De meme les possibilités de généralisation du lemme de Hensel à des anneaux plus généraux que Z[x] sont étudiées par Lauer (LA 83).

CHAPITRE 4 : FACTORISATION DES POLYNOMES A PLUSIEURS VARIABLES A COEFFICIENTS ENTIERS

1 - SCHEMA DE LA FACTORISATION

1.1 Préliminaires

P est un polynome à m+1 variables $x, x_1, \ldots, x_m$ à coefficients entiers de degré n en x qui est la variable principale , et de degré n_i en x_i.

La première étape de la factorisation consiste à se ramener à un polynome primitif sans carré. La méthode utilisée pour la factorisation à une variable reste valable en remplaçant les PGCD par des PGCD de polynomes à plusieures variables , et en remplaçant les dérivations par des dérivations partielles. D'autres sont possibles et peuvent etre plus efficaces , voir notament le papier de Wang et Trager (WA TR 79) .

La méthode de Kronecker pour la factorisation reste valable , mais déjà inefficace dans le cas d'une variable , elle l'est encore plus dans le cas de plusieur variables.

La méthode classique de factorisation d'un polynome à plusieures variables est très proche de la méthode de factorisation d'un polynome à une variable , et s'effectue en trois étapes.

1.2 Schéma de la factorisation

Etape 1 : Choisir m entiers "convenables" $, a_1, \ldots, a_m$ et factoriser $P(x, a_1, \ldots, a_m)$ sur $Z[x]$.

Etape 2 : Remonter les facteurs obtenus sur Z[x] en des facteurs modulo $(x_1-a_1)^{n1+1}, \ldots, (x_m-a_m)^{nm+1}$ à l'aide d'une version adaptée du lemme de Hensel.

Etape 3 : Reconstruire les facteurs vrais de P par recombinaison des facteurs obtenus et test de divisibilité.

1.3 Remarques

Choix des a_i : Les points d'évaluation $a_1, \ldots, a_m$ doivent satisfaire 2 conditions :

- Le degré de $P(x,a_1,\ldots,a_m)$ en x doit etre égal au degré de $P(x,x_1,\ldots,x_m)$ en x.

- Le polynome $P(x,a_1,\ldots,a_m)$ doit etre sans carré.

Factorisation de $P(x,a_1,\ldots,a_m)$: Si les a_i ne sont pas tous nuls , lès coefficients du polynome $P(x,a_1,\ldots,a_m)$ peuvent devenir très grands d'où l'interet de savoir factoriser sur $Z[x]$ des polynomes à gros coefficients , et donc l'interet de la factorisation modulo un nombre premier moyen qui réduit le temps de remontée des facteurs.

Borne sur les facteurs : Il existe deux sortes de bornes. Une borne porte sur le degré maximal que peuvent atteindre les facteurs du polynome, elle vaut ni pour la variable xi. L'autre borne porte sur la valeur absolue de tout coefficient d'un diviseur éventuel de P . Il est possible d'obtenir une telle borne à partir de celles données dans le cas une variables. Tous les calculs se feront modulo cette borne ou modulo une puissance de nombre premier la dépassant. Il sera donc possible de calculer l'inverse d'entiers si le nombre premier est bien choisi. On prend un nombre premier qui ne divise pas le coefficient de tete de P et tel que P soit sans carré modulo ce nombre premier. Malheureusement ces bornes sont très grandes , nous donnons celle de Gelfond , utilisée dans SAC2 .

$$M = \sum |a_i| \, 2^{(2^{n-1} + \sum 2^{ni-1} + 1)/2}$$

1.4 - Méthode de remontée

Présentation.

Il s'agit d'une méthode de remontée variable par variable , dérivée du lemme de Hensel. La factorisation de $P(x,a_1,\ldots,a_m)$ est en fait une factorisation de $P(x,x_1,\ldots,x_m)$ modulo $(x_1-a_1),\ldots,(x_m-a_m)$. Donc on va progresser variable après variable en passant d'une factoriston modulo $(x_i-a_i),\ldots,(x_m-a_m)$ à une factorisation $(x_{i+1}-a_{i+1}),\ldots,(x_m-a_m)$ jusqu'à atteindre la variable x_m. De plus la factorisation partielle concernant la i° variable est obtenue en remontant la factorisation modulo $(x_i-a_i)^i$ en la factorisation modulo $(x_i-a_i)^{i+1}$ jusqu'à atteindre n_i le degré de P en x_i.

exemple : $a_1 = a_2 = 0$ pour simplifier .

$$P(x,y,z) = x^3 + x^2(2 + y + yz + z^2 + z^3) + x(1 + 2y + y^2 + 2z^2 + y^2z + z^3y + z^5) + 2 + yz + y^2 + z^3 + y^3z + y^2z^3$$

Etape 1 : $P(x,0,0) = (x^2 + 1)(x + 2)$ qui est de degré 3 et sans carré.

Etape 2 : Remontée.

2.1 Remontée de y .

$$P(x,y,0) = (x^2 + 1 + xy + y^2)(x + 2) \bmod y^4$$

2.3 remontée de z.

$$P(x,y,z) = (x^2 + 1 + x(y + z^2) + y^2)(x + 2 + yz + z^3) \bmod (y^4, z^6)$$

Précision sur une étape de remontée. Prenons l'étape 2.2

i) $P(x,y,z) = (x^2 + 1 + xy + y^2)(x + 2) \bmod y4,z$

ii) remontée de z en z^2

$$P(x,y,z) = (x^2 + 1 + xy + y^2)(x + 2 + yz) \bmod y^4, z^2$$

iii) remontée de z^2 en z^3

$$P(x,y,z) = (x^2 + x(y + z^2) + 1 + y^2)(x + 2 + yz) \bmod y^4, z^3$$

iv) remontée de z^3 en z^4

$$P(x,y,z) = (x^2 + x(y + z^2) + 1 + y^2)(x + 2 + yz + z^3) \bmod y^4, z^4$$

Comme le degré de P en z est 5 il faut remonter le module en z jusqu'à z^6 , mais la dernière égalité est en fait une égalité sur $Z[x,y,z]$, il est donc inutile d'aller plus loin.

2 - ETAPE DE REMONTEE

2.1 Notations.

P est un polynome primitif sans carré de $Z[x,x_1,\ldots,x_m]$.

M est une borne majorant le double de la valeur absolue de tout facteur éventuel de P , choisie de la forme p^k , p étant un nombre premier convenable.

I est l'idéal engendré par $(x_1^{n1+1},\ldots,x_m^{nm+1})$

Ik est l'idéal engendré par $(x_1^{n1+1},..,x_k^{nk+1},x_{k+1},..,x_m)$

Ik,j est l'idéal engendré par $(x_1^{n1+1},..,x_{k-1}^{nk+1},x_k^{j},\ldots,x_m)$

On note P_k le polynome défini par :

$$P_k(x,x_1,\ldots,x_k) = P(x,x_1,\ldots,x_k,0,\ldots,0)$$

On suppose que $P(x,0,\ldots,0) = A(x)B(x)$, le polynome $P(x,0,\ldots,0)$ étant de degré n et sans carré. On suppose donc que les a_i de la première étape peuvent etre choisi tous nuls . Cela est fait afin d'exposer plus clairement l'étape de remontée. Dans le cas général , cela revient à supposer que le changement de variable $x_i \rightarrow x_i + a_i$ a été effectué. On verra plus tard comment éviter de réaliser ce changement de variable.

2.2 Lemme de Hensel.

On présente ici le lemme de Hensel série linéaire. L'approche série qui consiste à remonter les puissances successives d'une variable au lieu de doubler à chaque fois les puissances du module correspondant (méthode quadratique) , semble préférable ; voir Yun (YU 77) .

On verra plus loin la forme parallèle qui remonte tous les facteurs simultanément.

En fait la forme de remontée utilisée est très proche de la forme utilisée dans le cas une variable. En effet on remonte une égalité P = AB et une égalité AU + BV = 1 , mais au lieu de remonter le module q , on remonte les variables $x_1,\ldots,x_r$ suivant leurs puissances.

Il faut se rappeler que tous les calculs de la remontée se font modulo la borne M .

On pose A_0 et B_0 les polynomes égaux à A et B à un facteur multiplicatif entier près ,tels que : $P = A_0B_0 \mod M$, A_0 est unitaire. A_0 et B_0 sont des polynomes de $Z_M[x]$.

2.2.a Remontée d'une variable.

Il s'agit de remonter les puissances successives d'une variable.

On suppose que $P_k = Ak,_j(x,\ldots,x_k)Bk,_j(x,\ldots,x_k) \bmod Ik,j$,

c'est à dire que les puissances de x_k supérieures à j-1 ne sont pas prises en compte. Le problème est de remonter cette égalité de j à j+1. Le facteur Ak,j est supposé unitaire en x , ce qui permet de diviser par Ak,j.

De plus on suppose connus U_k et V_k polynomes de $Z[x,\ldots,x_{k-1}]$ tels que :

$$A_{k,1}(x,\ldots,x_k)U_k(x,\ldots,x_{k-1}) + B_{k,1}(x,\ldots,x_k)V_k(x,\ldots,x_{k-1}) = 1 \bmod I_{k,1}$$

et $\deg_x(V_k) \quad \deg_x(A_{k,j})$

Il s'agit en fait d'une égalité de Bezout entre polynomes de $Z[x,\ldots,x_{k-1}]$, car Ak,1 et Bk,1 sont en fait des polynomes de $Z[x,\ldots,x_{k-1}]$.

Proposition : Les polynomes $A_{k,j+1}$ et $B_{k,j+1}$ qui sont les facteurs cherchés de P modulo $I_{k,j+1}$ sont donnés par :

$A_{k,j+1} = A_{k,j} + x_k{}^j S_j$ et $B_{k,j+1} = P_k/A_{k,j+1} \bmod I_{k,j+1}$

avec $S_j = C.V_k \bmod A_{k,j}$ où C est donné par

$C = (P - A_{k,j}B_{k,j})/x_k{}^{j-1}$ modulo x_k.

De plus le polynome Ak,j+1 reste unitaire en x.

Preuve . Nécessairement $A_{k,j+1} = A_{k,j} + x_k S_j$

$B_{k,j+1} = B_{k,j} + x_k T_j$

avec S_j et T_j polynomes de $Z[x,\ldots,x_{k-1}]$, donc x_k ne figure pas dans T_j et S_j.

On veut que $P = A_{k,j+1}B_{k,j+1} \bmod I_{k,j+1}$

donc $$P - A_{k,j}B_{k,j} = x_k^{\,j}(A_{k,j}T_j + B_{k,j}S_j) \bmod I_{k,j+1}$$

$$C = A_{k,j}T_j + B_{k,j}S_j \bmod x_k$$

Cette égalité est donc une égalité dans $Z[x,\ldots,x_{k-1}]$, d'après des résultats sur les équations diophantiennes analogues à ceux du chapitre 3 , &1, d'après ce qu'on sait sur U_k et V_k on a donc: $S_j = CV_k \bmod A_{k,1}$.

Le résultat sur $A_{k,j+1}$ et $B_{k,j+1}$ s'en déduit immédiatement.

CQFD.

2.2.b Calcul des coefficients U_k et V_k

Ce calcul s'effectue à partir de l'égalité de Bezout dans Z[x] suivante : $A_0U + B_0V = 1 \bmod M$

A0 est égal à A multiplié par l'inverse dans Z_M du coefficient de tete de $P(x,0,\ldots,0)$.

Cette égalité peut se remonter de deux façons. La première consiste à construire une double suite $((U_0,\ldots,U_k),(V_0,\ldots,V_k))$ de polynomes tels que : U_i et V_i sont dans $Z[x,\ldots,x_i]$

$$U_iA_i + B_iV_i = 1 \bmod I_i$$

avec $A_i = A_{i\,ni+1}$ et idem pour B_i. Cette suite se construit de façon analogue à celle présentée en 2.2.a pour remonter les facteurs de P. On remonte U_{i-1} par puissances successives comme on a procédé pour $A_{k,i}$.

Une autre méthode consiste à calculer directement toutes les variables simultanément pour obtenir immédiatement U_k et V_k.

Soit J_i l'idéal engendré par les monomes $x_1^{\,i1}\ldots x_k^{\,ik}$ de degré total i , i.e. $i = \sum_{l=1 \text{ à } k} i_l$.

Supposons que $A_{k,j}U_i + B_{k,j}V_i = 1 \bmod J_i$

On cherche alors S_{i+1} et T_{i+1} de degré i en $x_1,\ldots,x_k$ tels que :

$$A_{k,j}(U_i + S_{i+1}) + B_{k,j}(V_i + T_{i+1}) = 1$$

i.e. $$S_{i+1}A_{k,j} + T_{i+1}B_{k,j} = C_{i+1}$$

Egalons les termes de degré total i en $x_1,\ldots,x_k$; on obtient :

$S_{i+1}A_0 + T_{i+1}B_0 = C'_{i+1}$ où C'_{i+1} est considéré comme un polynome de $A[x_1,\ldots,x_k]$ avec $A = Z_M[x]$.

Egalons les coefficients de chaque monome $x_1^{i1}\ldots x_k^{ik}$ pour chaque k-uplet $(i_1,\ldots,i_k)$, on trouve une série d'équations diophantiennes de $Z_M[x]$,

$$s_i(x)A_0 + t_i(x)B_0 = c_i(x)$$

qui permettent de calculer s_i et t_i à partir de l'équation $A_0U + B_0V = 1$.

On détermine ainsi S_{i+1} et T_{i+1}.

CQFD

2.2.c Remontée de toutes les variables

Il suffit pour remonter toutes les variables d'appliquer le résultat de 2.2.a sur la remontée d'une variable successivement, en partant de $P = A_0B_0$ avec A_0 unitaire en x, et de remonter chaque variable x_k jusqu'à la puissance n_k. Le résultat obtenu pour la variable x_k sert de point de départ pour remonter la variable x_{k+1} en posant : $A_{k+1,1} = A_{k,nk+1}$.

On obtient alors le résultat P = Am Bm où Am et Bm sont les facteurs de P modulo I et M . Cela permet d'obtenir la liste de tous les facteurs de P modulo I et M en les remontant les uns apès les autres , comme dans le cas une variable. Il est alors possible de passer à l'étape de reconstruction des facteurs , qui est semblable à l'étape de reconstruction du cas d'une variable.

2.3 Lemme de Hensel parallèle

Comme dans le cas une variable , il est possible de remonter tous les facteurs simultanément. L'algorithme et sa preuve sont très semblables à ce qui ce fait dans le cas une variable , aussi seules les grandes lignes de l'algorithme sont données ici. Tous les détails peuvent etre vus dans Wang et Rotschild (WA RO 76).

Algorithme de remontée parallèle.

Entrée : P polynôme primitif sans carré

Sortie : Liste des facteurs de P

(1) Choisir $a_1,\ldots,a_m$ convenables et factoriser $P(x,a_1,\ldots,a_m)$ effectuer le changement de variable $x_i \leftarrow x_i - a_i$. Les idéaux $I_{k,j}$ définis précédement se rapportent aux nouvelles variables.

(2) Factoriser a_n le coefficient de tete de P, qui est un polynome à m variables en utilisant l'algorithme de manière récursive. Calculer M la borne relative à P.

(3) $P = a_n R_1 \ldots R_r$ avec R_i poynomes unitaires premiers 2 à 2. On pose $R_{i,0} = R_i$ pour i = 1 à r. k=1 .

(4) Tant que k < m Faire
début
(on calcule $R_{k,i}$ tels que $P = a_n R_{k,1} \ldots R_{k,r} \bmod I_k$)

(4.1) $R_k{}^1{}_i = R_{k-1,i}$ pour i =1 à r.

(4.2) Calculer $A_{k,1},\ldots,A_{k,r}$ coefficients correcteurs tels que : $A_{k,1} W_{k,1} + \ldots + A_{k,r} W_{k,r} = 1 \bmod I_{k-1}$
où $W_{k,i} = \prod_{j=i} R_{k,j}$

(4.3) Pour i = 1 à n_k Faire
début
Remonter $R_k{}^j{}_i$ en $R_k{}^{j+1}{}_i$ par le sous algorithme décrit plus loin.
fin

(4.4) Pour i=1 à r , $R_{k,i} = R_k{}^{nk+1}{}_i$
fin

(5) Retranslater les facteurs $R_{m,i}$ obtenus en faisant le changement de variable $x_i \leftarrow x_i - a_i$; obtenir les facteurs vrais de P à l'aide des facteurs unitaires $R_{m,i}$ et des facteurs du coefficient de tete.

fin

Sous algorithme de calcul des $R_k{}^{j+1}{}_i$ à partir des $R_k{}^j{}_i$

Entrée : Liste des $R_k{}^j{}_i$ et liste des coefficients correcteurs $A_{k,i}$

Sortie : Liste des $R_k{}^{j+1}{}_i$

(1) $C = (P_k - an(R_k{}^j{}_1 \ldots R_k{}^j{}_r)) \bmod I_{k,j})/ x_k{}^j$

(2) Pour i = 1 à r Faire

début

(2.1) Calculer S_{ij} solution de $C = \sum S_{ij} W_{ki}$
avec $W_{ki} = \prod_{j \neq i} R_{k-1,j}$
en posant $S_{ij} = (C.A_{k,i} \mod(x_k,\ldots,x_m)) \mod R_{k-1,i}$

(2.2) $R_k{}^{j+1}{}_i = R_k{}^j{}_i + x_k{}^j S_{ij}$

fin

(3) Sortir la liste des $R_k{}^{j+1}{}_i$

fin

Remarques : - Les divisions par $R_{k-1,i}$ ne posent pas de problème car ces polynomes sont unitaires.

- Le calcul des coefficients correcteurs se fait par un algorithme semblable à celui décrit en 2.2.b pour 2 facteurs. Le point de départ sera l'identité de Bezout :

$$\sum_{i=1 \text{ à } r} A_{i0} \prod_{j \neq i} R_{j0} = 1 \text{ dans } Z_q[x]$$

obtenue à l'aide de l'algorithme d'Euclide ou de la décomposition en éléments simples de la fraction $1/ R_1 \ldots R_r$.

3 - PROBLEMES GENERAUX

3.1 Facteurs parasites.

Comme dans le cas une variable, la réduction de $P(x,x_1,..,x_m)$ en $P(x,a_1,..,a_m)$ peut introduire des facteurs parasites. Si le polynome P a m r facteurs sur $Z[x,x_1,\ldots,x_m]$, le polynome $P(x,a_1,\ldots,a_m)$ a au moins r facteurs sur $Z[x]$, mais peut en avoir plus. Un facteur du polynome réduit qui ne donne pas un facteur du polynome de départ est dit facteur parasite.

L'existence de ces facteurs parasites complique les calculs de remontée et impose une étape de reconstruction des facteurs qui peut etre exponentielle en le degré du polynome en x.

3.2 Coefficient de tête

Dans la méthode de remontée proposée, les facteurs sont unitaires, donc meme s'il n'y a pas de facteur parasite, les facteurs remontés ne sont égaux à un vrai facteur de P qu'à un facteur du coefficient de tete près.

Si le coefficient de tete du polynome vaut 1 ou si c'est un entier, il n'y a pas de problème car on peut remonter les facteurs obtenus dans Z x en gardant leur coefficient de tete. Si ce n'est pas le cas les calculs sont plus complexes, d'après la proposition qui suit.

Proposition : Soit $Q(x_1,\ldots,x_m)$ un polynome de $Z[x_1,\ldots,x_m]$ avec

$$Q(x_1,\ldots,x_m) = 1 - S(x_1,\ldots,x_m) \text{ et } S(0,\ldots,0) = 0.$$

Alors Q est inversible dans $A[x_1,\ldots,x_m] / I$ où I est l'idéal engendré par $(x_1^{n_1},\ldots,x_m^{n_m})$ et son inverse vaut :

$$Q^{-1} = 1 + S + S^2 +\ldots+ S^{n-1} \quad \text{si n est le maximum des } n_i.$$

Cette proposition d'algèbre bien connue découle du fait que $S^n = 0$ dans le quotient de $A\ x_1,\ldots,x_m$ par I .Le calcul montre qu'alors QQ^{-1} vaut 1 .

exemple : $Q(y,z) = 1 - y - z$ avec $I = (y^4,z^4)$

$$Q^{-1}(y,z) = 1 + y + z + (y + z)^2 + (y + z)^3$$

Si Q est le coefficient de tete d'un facteur de P , le facteur unitaire correspondant est égal à son produit par Q^{-1}. L'exemple permet d'apprecier l'explosion du nombre de termes entrainée par le fait de ne remonter que les facteurs unitaires et par suite l'importance du problème du coefficient de tete.

3.3 Problème des mauvais zéros.

Le choix des a_i de l'étape 1 de la factorisation oblige ensuite à effectuer le changement de variable $x_i \to x_i + a_i$. Si les a_i ne sont pas nuls, cela peut amener des calculs très couteux car on doit calculer le polynome $Q(x,y_1,\ldots,y_m)$ qui vaut $P(x,x_1+a_1,\ldots,x_m+a_m)$. Cela impose de developper des termes du type $(x_1+a_1)^{n_1}\ldots(x_m+a_m)^{n_m}$ qui prennent beaucoup de temps de calcul, meme avec un polynome de départ simple.

exemple : $P(x,y,z) = x^2y^{10}z^{20} - 1$ qui se factorise à la main en

$(xy^5z^{10} - 1)(xy^5z^{10} + 1)$ va demander le calcul de

termes en $(y+1)^{10}(z+1)^{20}$.

On cherche donc à prendre le maximum de a_i nuls , mais cela n'est pas toujours possible comme le montre l'exemple ci-dessus , car les a_i doivent satisfaire :

- Ils n'annulent pas le coefficient de tete de P
- $P(x,a_1,\ldots,a_m)$ est sans carré.

Quand il n'est pas possible de choisir les a_i tous nuls on est confronté au problème des mauvais zéros.

4 - SOLUTIONS CLASSIQUES

4.1 Facteurs parasites.

Contrairement au cas une variable , il n'existe pas de polynome irréductible de $Z\ x,x_1,\ldots,x_m$ réductible dans $Z\ x$, après la transformation $x_i - a_i$ pour tous les choix des a_i Cela est du au théorème d'irréductibilité de Hilbert.

Théorème (Hilbert) : Soit $P(x,x_1,\ldots,x_m)$ un polynome irréductible de $Z[x,x_1,\ldots,x_m]$, soit N(n) le nombre de m-uplets $(a_1,\ldots,a_m)$ tels que $|a_i| < n$ pour i de 1 à m et tels que $P(x,a_1,\ldots,a_m)$ soit réductible dans Z[x] ; alors il existe 2 constantes C et d dépendant de P telles que :

$$0 < d < 1$$

$$N(n) < C\ (2n + 1)^{m-d}$$

Cependant il n'existe pas d'estimations raisonnables sur C et d ce qui empêche toute exploitation pratique de ce résultat.

La méthode classique, suggérée par Wang (WA 78) est de calculer plusieures factorisations $P(x,a_1,\ldots,a_m)$ pour plusieurs choix des a_i et de garder le m-uplet qui donne un nombre minimal de facteurs. En pratique cette méthode heuristique s'avère assez efficace.

Une méthode rigoureuse serait d'utiliser l'algorithme de Lenstra (LE 82) pour le cas plusieures varables. Cette façon de procéder ne semble pas retenue à cause du cout de l'algorithme de construction d'une base réduite dans un treillis.

4.2 Coefficient de tete.

Le problème du coefficient de tete complique beaucoup les calculs lors de la remontée , aussi il est interessant de calculer à l'avance les coefficients de tete des facteurs du polynome.

Supposons qu'il n'y a pas de facteur parasite donc chaque facteur sur Z[x] donne un vrai facteur de P . Cette supposition est raisonnable d'une part d'après le théorème de Hilbert , d'autre part d'après l'expérience qui montre que le choix de plusieurs m-uplets différents élimine presque entierement ce risque.

Supposons connue la factorisation du coefficient de tete A_n du polynome P.

$A_n = b\ F_1^{n1} \ldots F_l^{nl}$, b un entier , F_i les facteurs irréductibles de A_n.

On suppose que les a_i satisfont les trois conditions :

(i) $A_n(x,a_1,\ldots,a_m) = 0$

(ii) $P(x,a_1,\ldots,a_m)$ est sans carré

(iii) Pour tout F_i , $F_i(a_1,\ldots,a_m)$ a au moins un diviseur premier p_i qui ne divise aucun des $F_j(a_1,\ldots,a_m)$ pour $j<i$, et qui ne divise ni b , ni le PGCD des coefficients de $P(x,a_1,\ldots,a_m)$ noté d.

La troisième condition va permettre de résoudre le problème du coefficient de tete. De plus cette condition n'est pas trop restrictive et les m-uplets qui satisfont les trois conditions forment un ensemble dense.

Proposition : Si $P(x,a_1,\ldots,a_m)$ se factorise en $d.r_1\ldots r_r$ sans facteur parasite, avec r_i polynomes irréductibles , si P se factorise en $R_1\ldots R_r$ sur $Z[x,x_1,\ldots,x_m]$,le coefficient de tete de R_i étant T_i et t_i celui de r_i , alors :

F_j^k divise $T_i = (F_j(a_1,\ldots,a_m))^k$ divise dt_i

(d est le PGCD des coefficients de $P(x,a_1,\ldots,a_m)$

Preuve. Si F_j^k divise R_i alors de façon immédiate, $F_j(a_1,\ldots,a_m)^k$ divise $d_i t_i$ si d_i est un diviseur de d.

Réciproquement , si F_i^k ne divise pas T_i , alors $F_i(a_1,\ldots,a_m)^k$ ne divise pas $d_i F_1^{n1}(a_1,\ldots,a_m)\ldots F_k^{nk}(a_1,\ldots,a_m)$ avec $k \neq i$, qui vaut T_i , car p_i ne divise aucun des $F_l^{nl}(a_1,\ldots,a_m)$ pour $l<i$.

Il est donc possible de distribuer tous les facteurs F_i^k en commençant par F_1, puis $F_{l-1},\ldots$, jusqu'à F_1. Il reste alors à distribuer les facteurs b_i de b. On rappelle que A_n le coeffcient de tete se factorise en $bF_1^{n1}\ldots F_I^{nI}$.

$Ti = b_i D_i$ avec D_i un produit des F_i à certaines puissances. Si le polynome reste primitif après sa réduction sur $Z[x]$, i.e. si $d = 1$, alors $b_i = t_i/D_i(a_1,..,a_m)$.

Si $d = 1$, soit on essaie de calculer b_i à l'aide d'un calcul de PGCD sans etre sur de réussir , soit on factorise le polynome $d^{r-1}P$ dont les facteurs ont pour coefficient de tete $d^{r-1}D_i$. Dans ce cas il suffit de rendre primitifs les facteurs trouvés pour obtenir les facteurs de P.

exemple : $P(x,y,z) = x^2(6 + 4y + 3z + 2yz) + x(5 + 5y + 3z + z^2)$

$$+ 1 + y + z + yz$$

Le couple (0,0) satisfait les trois conditions.

$$P(x,0,0) = 6x^2 + 5x + 1 = (3x + 1)(2x + 1)$$

Le coefficient de tete se factorise en : $(3 + 2y)(2 + z)$

Pour $y = z = 0$, $3 + 2y = 3$ qui divise le coefficient de tete de $3x + 1$, alors que 2 ne le divise pas. De plus 2 divise le coefficient de tete de $2 + z$; par suite le coefficient de tete du facteur de P qui donne $3x + 1$ est $3 + 2y$ et celui du facteur qui donne $2x + 1$ est $2 + z$.

Remarque : S'il y a un facteur parasite , on ne trouve pas les coefficients de tete des facteurs , mais les calculs de remontée restent valables dans l'anneau quotient $Z[x,x_1,\ldots,x_m]/I$

4.3 Problème des mauvais zéros

Pour résoudre les deux problèmes précédents , on a été amené à considérer des factorisations $P(x,a_1,\ldots,a_m)$ où il n'est pas possible d'imposer un nombre maximal des a_i nuls. Il devient nécessaire d'avoir une méthode qui résolve de manière efficace le problème des mauvais zéros. En particulier il faut éviter de calculer effectivement le changement de variable $x_i \rightarrow x_i + a_i$.

La difficulté est alors de calculer un polynome $C(x,x_1,..,x_k)$ modulo $(x_k-a_k)^I$. On utilise pour cela une dérivation partielle.

Nous présentons l'algorithme proposé par Wang (WA 79) qui résoud les trois problèmes abordés en ne developpant que les points nouveaux.

Les notations utilisées précédement restent valables en remplaçant x_i par x_i-a_i pour les idéaux. On introduit J l'idéal engendré par$(x_1^{n1+1},\ldots,x_m^{nm+1})$

Factorisation d'un polynome primitif sans carré.

(1) Choisir $a_1,\ldots,a_m$ satisfaisant les conditions de 4.3 et factoriser $P(x,a_1,\ldots,a_m)$. Effectuer plusieures fois cette opération et garder le meilleur choix. Calculer la borne M.

(2) Calculer les coefficients de tete des facteurs de P

(3) Calculer les R_{ki} tels que
   (a) $P = R_{k1}\cdots R_{kr} \bmod(I_k,J,M)$

   (b) $R_{ki} = r_i \bmod (x_1-a_1,\ldots,x_m-a_m)$ si les r_i sont les facteurs de $P(x,a_1,..,a_m)$

   (c) Le coefficient de tete de R_{ki} est égal modulo I_k à celui du facteur de P relatif à r_i.

fin

Calcul des Rki.

Les R_{ki} se calculent à l'aide d'une suite auxiliaire $R_k^j{}_i$ pour j de 1 à $n_k + 1$. Cela correspond à la remontée des facteurs modulo $(x_k-a_k)^{J_k}$ en facteurs modulo $(x_k-a^k)^{J+1}$.

On a $R_k^i{}_1 = R_{k-1i}$ et $R_k^{nk+1}{}_i = R_{ki}$

(a) Calculer les coefficients correcteurs A_{ki} tels que :

$$A_{k1} W_{k-1}{}_1 + \ldots + A_{kr} W_{k-1r} = 1 \bmod(I_k, M)$$
$$\deg(A_{ki}) < \deg(R_{ki})$$

avec $W_{k-1i} = \prod_{j \neq i} R_{k-1j}$

(b) Posons $C = P_k - \prod R_k{}^j{}_i$

on rappelle que $P_k(x, x_1, \ldots, x_k) = P(x, x_1, .., x_k, a_{k+1}, .., a_m)$

Soit $C_j = 1/j!(D^j C/Dx_k{}^j)_{(xk=ak)}$. C_j est la j° dérivée partielle par rapport à la variable x_k, évaluée en $x_k = a_k$.

Proposition :

$R_k{}^{j+1}{}_i = R_k{}^j{}_i + (x_k - a_k)^j S_{ij}$ avec S_{ij} polynome de $Z[x, x_1, \ldots, x_{k-1}]$ tel que :

$S_{ij} = A_{ki} C_j$ modulo R_{k-1i}.

Preuve. $R_k{}^{j+1}{}_i$ est nécessairement de la forme $R_k{}^j{}_i + (x_k - a_k) S_{ij}$.

$P_k = P_k{}^{j+1}{}_1 \ldots P_k{}^{j+1}{}_r$ conduit à l'égalité:

$$P_k - R_k{}^j{}_1 \ldots R_k{}^j{}_r = (x_k - a_k)^j \left(\sum_{i=1}^{} S_{ij_r} W_{k-1i} \right) \bmod (x_k - a_k)^{j+1}$$

Le premier terme de l'égalité est le polynome C défini précédement. Il faut donc calculer les polynomes S et D tels que: $C = (x_k - a_k)^j S + D$ où S est un polynome de Z $x, x^1, \ldots x^{k-1}$ et où D est un polynome de $Z[x, x_1, \ldots, x_k]$, nul modulo(xk -ak)j+1.

D'après la formule de Taylor appliquée au polynome C en la variable x_k, on a que :

$$S = 1/j!(D^j C/Dx_k{}^j)_{(xk=ak)}$$

Alors les S_{ij} vérifient $S = \sum S_{ij} W_{k-1i}$, le degré de S en x étant inférieur à celui de P à cause de la détermination exacte du coefficient de tete, donc on le résultat:

$S_{ij} = A_{ki} S$ modulo R_{k-1i}.

CQFD

Calcul des coefficients correcteurs A_{ki}.

A la première itération ,il s'agit de polynomes de Z[x] , calculés par l'algorithme d'Euclide ou celui de Kung et Tong.

Il suffit alors de remonter cette égalité, soit variable par variable, soit simultanément comme on l'a exposé en 2.2.c .Dans ce cas la difficulté est d'évaluer les coefficients de monomes de degré i en $(x_1-a_1)^{i1}\ldots(x_k-a_k)^{ik}$. Cela peut se faire soit directement, soit en calculant des dérivées partielles et en les évaluant aux points ai.

Mais aussi bien dans la remontée des facteurs que dans les calcul des coefficients correcteurs , on évite d'effectuer de changement de variable $x_i \leftarrow x_i - a_i$, ce qui est possible grace à l'usage des dérivations partielles ,suivies d'évaluation aux points adéquats. Une description plus précise de cet algorithme se trouve dans Wang (WA 79).

5 - UTILISATION DE LA DECOMPOSITION D'UNE FRACTION RATIONNELLE

5.1 L'idée

Viry (VI 82) propose un algorithme de remontée qui utilise les fractions rationnelles. Malheureusement cet algorithme nécessite un polynome de départ unitaire et unitarise les polynomes non unitaires à l'aide du changement de variables $x_i \leftarrow x_i + p_i x$. Ce changement augmente de façon considérable le dégré du polynome en x, donc le cout de la factorisation sur Z[x] et la possibilité d'obtenir des facteurs parasites. Un des points de ce travail est de retenir les éléments interessants de cet algorithme et de remédier à ses inconvénients. Nous avons obtenu un nouvel algorithme de remontée basé sur les fractions rationnelles et un tout nouveau algorithme pour réaliser le changement de variable classique xi ← xi - ai.

L'idée essentielle est de remonter toutes les variables simultanément et de calculer les nouvelles approximations des facteurs à l'aide des fractions rationnelles.

Si $P = A_i B_i$ à l'itération i alors $P = A_{i+1} B_{i+1}$ avec
$A_{i+1} = A_i + A_i'$
$B_{i+1} = B_i + B_i'$

On calcule Ai' et Bi' comme solutions de la décomposition en éléments simples de la fraction $(P - A_i B_i)/A_0B_0$ où A_0 et B_0 sont les facteurs sur Z[x].

Il suffit de calculer les décompositions de xj/A0B0 sur Z[x], pour $0 \leqslant j < n$ le degré de P en x.

L'algorithme est décrit en 5.2 et l'algorithme de changement de variable pour résoudre le problème des mauvais zéros en 5.5.

5.2 Notations

$P(x,x_1,\ldots,x_m)$ est un polynome primitif sans carré à coefficients entiers.

Le degré de P en x est n , le degré en x_i est n_i.

M est une borne sur les coefficients des facteurs de P.

p est le degré global de P en $x_1,\ldots,x_m$ i.e. c'est le degré du monome de plus fort degré en $x_1,\ldots,x_m$.

I_i est l'idéal engendré par les monomes homogènes de degré i en $y_1,\ldots,y_m$ dans $A[x_1,\ldots,x_m]$ avec A = Z[x], et $y_i = x_i - a_i$.

Par exemple si $P(x,x_1,x_2,x_3) = x^2x_1 + xx_2^4x_3 + x_1^3 + 2$ alors n = 2 , n1 = 3 , n2 = 4, n3 = 1 , p = 5

Si $a_1=a_2=1$, $a_3=2$ et i = 2 alors I_2 est l'idéal engendré par
$((x_1-1)^2,(x_2-1)^2,(x_3-2)^2,(x_1-1)(x_2-1),(x_2-1)(x_3-2),(x_3-2)(x_1-1))$

5.3 L'algorithme.

On présente l'algorithme série sans prédétermination des coefficients de tete, les autres versions seront vues en 5.4.

Algorithme de factorisation .

Entrée : P polynome primitif sans carré.

Sortie : Liste des facteurs de P

(1) Choisir

$a_1,\ldots,a_m$ et factoriser $P(x,a_1,\ldots,a_m)$

(2) Calculer M et remonter l'égalité P = A B , A étant un facteur unitaire de P sur ZM[x] . Après remontée poser P = P/A et remonter les autres facteurs jusqu'à épuisement de ceux-ci.

(3) Reconstruire les facteurs vrais de P.

Algorithme de remontée d'un facteur de P

On va remonter toutes les variables simultanément.

(1) (Initialisation) Calculer P_i pour i = 1 à p, P_i est la partie homogène de degré i en $y_1=x_1-a_1,\ldots,y_m=x_m-a_m$ à l'aide de l'algorithme de changement de variable.
On calcule en meme temps Ai la partie homogène de degré i du coefficient de tete de P.

(2) i=0 , $H_0 = A$, $Q_0 = B$ où $P = A\ B \bmod(M,I_1)$ et A unitaire.
on rappelle que $I_1 = (y_1,\ldots,y_m)$

(3) Calculer la décomposition en fraction rationnelles de

x^j/H_0Q_0 pour $0 \leqslant j < n$.

$Q_0' = A_1x^l$ avec l = deg(Q) correction due au coefficient de tete.
i=1 , $H_1 = H_0$, $Q_1 = Q_0 + Q_0'$

(4) Tant que $i \leqslant p$ Faire

début

(4.1) $S_i = (P - H_iQ_i)_i$ partie homogène de degré i du polynome $P - H_iQ_i$

(4.2) Calculer H_{i+1}'' et Q_{i+1}'' solutions de la décomposition en éléments simples de S_i / H_0Q_0 à l'aide des décompositions x^j/Q_0H_0

(4.3) $Q_{i+1}' = A_{i+1}\ x^l$ correction pour le coefficient de tete.

(4.4) $H_{i+1} = H_i + H_{i+1}''$; $Q_{i+1} = Q_i + Q_{i+1}' + Q_{i+1}''$; i = i+1

fin

fin

Remarques - Calcul de S_i partie homogène de degré i de $P - H_iQ_i$. Aucun calcul supplémentaire n'est à effectuer car:

$$S_i = (P - H_iQ_i)_i = P_i - \sum_{\substack{j+k=i \\ j+k=1}} H_j Q_k$$

où les H_j et Q_k sont les parties homogènes de degré respectifs j et k de H_j et Q_k qui sont calculées au cours de l'algorithme car $H_j = H_{j+1}''$ et $Q_k = Q_{k+1}'' + Q_k'$

- Calcul des décompositions x^j/Q_0H_0. Elles se calculent par l'algorithme de Kung et Tong dans Z_M x pour $j = 0$. Ensuite elles se déduisent aisément par multiplications successives par x et le calcul de $x^{deg(H0)} \bmod H_0$ et idem pour Q_0.

- Calcul des décompositions en éléments simples de l'étape 4.2 . Il faut calculer la décomposition de S_i/H_0Q_0. On sait que le degré de S_i en x est inférieur à celui de P à cause de la correction faite pour le coefficient de tete. Alors

$$S_i/H_0Q_0 = \sum_{0 \leqslant j < n} G_{ij}\, x^j/H_0Q_0 \quad \text{où } G_{ij} \text{ est un élément de } Z[x_1,..,x_m]$$

Il suffit de connaitre les décompositions des x^j/H_0Q_0 pour calculer celle de S_i/Q_0H_0.

Preuve de l'algorithme.

Il suffit de montrer qu'à chaque étape de remontée ,

$P = H_iQ_i \bmod I_i$

$H_i = H_0 \bmod I_1$ et $Q_1 = Q_0 \bmod I_1$

C'est vrai à l'itération 1

Supposons le vrai à l'itération i.

Supposons que $P = H_{i+1}Q_{i+1} \bmod I_{i+1}$ alors $H_{i+1} = H_i + U_i$ et $V_{i+1} = Q_i + V_i$ avec U_i et V_i de degré i+1 en $y_1,...,y_m$.

Donc modulo I_{i+1} on obtient:

$$P - H_i Q_i = H_iVi + Q_iU_i$$

Identifions les termes homogènes de degré i+1 en $y_1,...,y_m$, il

arrive :

$$(P - H_i Q_i)_{i+1} = H_0 V_i + Q_0 U_i$$

Grace à la correction sur le coefficient de tete relative à Q_i le premier terme est un polynome de degré en x inférieur à n. De plus , considérés comme polynomes en x , U_i et V_i ont des degrés inférieurs à ceux de H_0 et Q_0 respectivement.

Donc l'égalité obtenue définit U_i et V_i comme solutions de la décomposition en éléments simples de S_{i+1}/H_0Q_0 . Cette décomposition est possible dans $Z_M[x_1,\ldots x_m]$ x car H_0 est unitaire. De plus M étant convenablement choisie, il est possible d'inverser les entiers qui doivent l'etre dans Z_M.

La correction ajoutée à Q_i étant nulle modulo I_{i+1} , on a bien l'égalité cherchée. Les relations entre H_i et H_0 , Q_i et Q_0 sont évidentes par construction.

CQFD

Exemple : $P(x,y,z) = x^2(6 + 4y + 3z + 2yz)$

$+ x(5 + 5y + 3z + z_2 + 2y_2)$

$+ 1 + y + z + yz$

$a_1 = a_2 = 0$ et $P(x,0,0) = (3x + 1)(2x + 1)$

On prend $M = 17^2$

$H_0 = x + 145$ et $Q_0 = 6x + 2$.

$1/H_0Q_0 = -1/H_0 + 6/Q_0$

$x/H_0Q_0 = 145/H_0 + -2/Q_0$

Le coefficient de tete est $6 + 4y + 3z + 2\,yz$ donc

$Q_1 = (6 + 4y + 3z)x + 2$

Etape 1 : $S_1 = (P - Q_0H_0)_1 = x(3y + 146z) + y + z$

d'où $H_2' = 145y + 72z$ et $Q_2' = 3z$;

donc

$$H_2 = x + 145 + 145y + 72z$$

$$Q_2 = x(6 + 4y + 3z + 2yz) + 2 + 3z$$

Etape 2 : $S_2 = (P - H_1 Q_1)_2 = x(74z2 + 143yz) + 73z^2 + 144yz$ ce qui donne :

$$H_3 = x + 144 + 144y + 72z - 36z^2 + 72yz$$

$$Q_3 = x(6 + 4y + 3z + 2yz) + 2 + 3z + z_2$$

Le degré global p de P en x_1, x_2 étant 2 , on passe à la reconstruction des facteurs qui donne $H = x(2 + z) + 1 + y$ et $Q = x(3 + 2y) + 1 + z$.

5.4 Versions possibles de l'algorithme.

Comme pour l'algorithme classique , il est possible de proposer deux versions de l'algorithme :

- Une version avec prédétermination des coefficients de tete des facteurs .

- Une version qui remonte simultanément tous les facteurs.

5.4.1 Utilisation du calcul des coefficients de tete.

On suppose qu'il n'y a pas de facteurs parasites et que les coefficients des facteurs sont calculés , par exemple par la méthode donnée en 4.2.

Le seul changement à apporter à l'algorithme est d'introduire les corrections dues à ces termes soient H'_i et Q'_i à la i° itération de façon qu'à l'itération suivante le coefficient de $(P - H_i Q_i)$ soit nul.

On posera donc après la décomposition en éléments simples,
$H_{i+1} = H_i + H_{i+1} + H_i'$
$Q_{i+1} = Q_i + Q_{i+1} + Q_i'$
et cela remplace les étapes 4.2 et 4.3 de l'algorithme.

L'intéret du précalcul des coefficients de tete est que, s'il n'y a pas de facteurs parasites, les égalités modulo M et I_{i+1} sont des égalités dans Z , qu'il n'y a pas de facteurs modulo M^{i+1}. Donc les coefficients ont une taille raisonnable puisque ce sont les vrais coefficients des facteurs.

5.4.2 Remontée parallèle.

Comme dans le cas une variable , il est possible de remonter tous les facteurs simultanément à l'aide du nouvel algorithme. Le schéma de remontée , ici avec prédétermination des coefficients de tete, est le suivant:

Remontée simultanée de tous les facteurs.

(1) Calculer les R_j facteurs sur Z de $P(x,a_1,..,a_m)$. On pose $R_{j0} = R_j$ pour j=1 à r .

(2) Calculer les coefficients de tete des facteurs de P sur $Z[x,x_1,...,x_m]$.

(3) Calculer les décompositions en éléments simples des fractions $x^j/R_{10}...R_{r0}$ pour j de 0 à n-1.

(4) $R_{j1} = R_{j0} + R'_{j0}$ pour j = 1 à r et avec R'_{j0} correction relative au coefficient de tete pour le j^{o} facteur. i=1.

(5) Tant que $i \leqslant p$ Faire

début

(5.1) $S_i = (P - R_{1i}...R_{ri})_i$ partie homogène de degré i du polynome entre parenthèse.

Pour j=1 à r Faire

début

(5.2) Calculer $R_j"_{i+1}$ solution de la décomposition en éléments simples de $S_i /R_{10}...R_{r0}$.

(5.3) Calculer R'_{ji+1} correction relative au coefficient de tete

(5.4) $R_{ji+1} = R_{ji} + R_j"_{i+1} + R'_{ji+1}$

fin

fin

(5) Reconstruire les facteurs

fin

La validité de l'algorithme se montre de façon analogue à celle de l'algorithme série.

Le calcul de la partie homogène de degré i se fait par :

$$(P - R_{1i}\cdots R_{ri})_i = P_i - \sum_{l_1+\ldots+l_r = i} R'_{1i}{}^{l_1}\cdots R'_{ri}{}^{l_r}$$

avec $R_{ji}{}^{l_j}$ partie homogène de degré l_j de R_{ji}.

6 - ALGORITHME DE CHANGEMENT DE VARIABLES

On présente ici un algorithme qui calcule P_i la partie homogène de degré i en $y_1=x_1-a_1,\ldots,y_m=x_m-a_m$ d'un polynome P pour i de 1 à p , le degré global du polynome en les y_i ,qui est aussi le degré global du polynome en les x_i.

La méthode permettant d'effectuer facilement le changement de variable $x_i=y_i+a_i$ une fois les P_i connus sera ensuite décrite.

6.1 Calcul des parties homogènes.

Dans ce qui suit le polynome $P(x,x_1,\ldots,x_m)$ sera considéré comme un polynome de $A[x_1,\ldots,x_m]$, la variable x n'apparaitra plus.

Alors $P = P_0 + P_1 + \ldots + P_p$ avec P_i partie homogène de degré i en $y_1,\ldots,y_m$ avec $y_i = x_i - a_i$.

Immédiatement on a : $P_0 = P(a_1,\ldots,a_m)$, donc nous supposerons dorénavant que les polynomes considérés satisfont $P_0 = 0$, sinon il suffira de prendre $P = P - P_0$.

exemple : $P(x_1,x_2) = x_1^2 - 2x_1x_2 + 2x_2 + x_1 + - 1$

$y_1 = x_1 - 1$ et $y_2 = x_2 - 1$ alors p = 2 et

$P = P_0 + P_1 + P_2$ avec $P_0 = 1$, $P_1 = y_1 + y_2$, $P2 = y_1^2 - 2y_1y_2$

On présente d'abord les propriétés mathématiques qui fondent

l'algorithme proposé.

La base est l'identité d'Euler : Si Q est un polynome homogène de degré k en $x_1,\ldots,x_m$ alors

$\sum_{l=1 \text{ à } m} x_l \, DQ/Dx_l = k\,Q$ où DQ/Dx_l est la dérivation partielle par rapport à la l° variable.

Comme $y_l = x_l - a_l$ on a que $DQ/Dx_l = DQ/Dy_l$, ce qui permet d'écrire:

$$\sum_{l=1 \text{ à } m} (x_l - a_l) DP/Dx_l = \sum_{i=1 \text{ à } p} i\, P_i$$

En itérant le procédé , on arrive au système d'équations linéaire en P_i suivant :

$$P_1 + P_2 + \ldots + P_p = G_0$$
$$P_1 + 2P_2 + \ldots + pP_p = G_1$$
$$\vdots$$
$$P_1 + 2^{p-1}P_2 + \ldots + p^{p-1}P_p = G_{p-1}$$

Avec $G_0 = P$ et $G_j = \sum_{l=1 \text{ à } m} (x_l - a_l)\, DG_{j-1}/Dx_l$

Donc les P_i sont donnés par les formules classiques de Cramer, $P_i = D_i / D$ où les D_i et D sont les déterminants habituels.

Posons $b_l = l$ pour i=1 à p. Alors D est le déterminant de Van der Monde suivant :

$$D = \begin{vmatrix} 1 & \cdots & 1 \\ b_1 & & b_p \\ \vdots & & \vdots \\ b_1^{p-1} & & b_p^{p-1} \end{vmatrix} = \prod_{1 \leqslant k < l \leqslant p} (b_l - b_k)$$

Et D_i est obtenu en substituant à la i° colonne de D la colonne formée des G_j pour j = 0 à p-1. Donc le coefficient de b_i

dans D_i est le coefficient de b_i dans D . Or on sait que D vaut

$$\prod_{j\neq i} (b_i - b_j) \prod_{l,k \neq i \text{ et } 1\leq k<l\leq p} (b_l - b_k)$$

Si $R(X) = \prod_{j=1 \text{ à } p} (X - j)$ et $R_i(X) = R(X)/(X-i)$ alors : $P_i = A_i/r_i$ où A_i est le polynome obtenu en substituant G_j à X^j dans $R_i(X)$ et où $r_i = R_i(i)$.

L'algorithme de calcul des parties homogènes de P est le suivant :

Calcul des parties homogènes .

Entrée : P polynome , liste $(a_1,\ldots,a_m)$

Sortie : Liste des parties P_i homogènes de degré i en $y_1,\ldots,y_m$

(1) $P_0 = P(a_1,\ldots,a_m)$, $P = P - P_0$, Calcul du degré p de P.

(2) Pour j=0 à p-1 calculer G_j par $G_0 = P$ et $G_j = \sum_{l=1 \text{ à } m} (x_l - a_l) DP/Dx_l$.

(3) Calculer $R(X) = \prod_{j=1 \text{ à } p} (X - j)$ et $R_i(X) = R(X)/(X-i)$

(4) Pour i=1 à p calculer $P_i = A_i/r_i$ avec A_i obtenu en substituant G_j à X^j dans R_i , et $r_i = R_i(i)$.

fin

Exemple : Pour simplifier on prend un polynome de $Z[x_1,x_2]$, c'est à dire que A se réduit à Z.

$P(x_1,x_2) = 3 + 6x_1 + x_2 + x_1x_2 + 4x_1^2 + x_1^3$

avec $y_1 = x_1+1$ et $y_2 = x_2+1$ c'est à dire que $a_1=a_2=-1$.

$P(-1,-1) = 0$

Le calcul des Gi donne:

$$G_0 = 3 + 6x_1 + x_2 + x_1x_2 + 4x_1^2 + x_1^3$$

$$G_1 = 7 + 15x_1 + 2x_2 + 2x_1x_2 + 11x_1^2 + 3x_1^3$$

$$G_2 = 17 + 39x_1 + 4x_2 + 4x_1x_2 + 31x_1^2 + 9x_1^3$$

Le calcul des R_i donne : $R_1(X) = X^2 - 5X + 6$

$$R_2(X) = X^2 - 4X + 3$$

$$R_3(X) = X^2 - 3X + 2$$

et r1 = 2 , r2 = -1 , r3 =2 .

On trouve $P_1 = 0$, $P_2 = x_1^2 + x_1x_2 + 3x_1 + x_2 + 1$,

$P_3 = x_1^3 + 3x_1^2 + 3x_1 + 1$.

6.2 Changement de variable effectif.

Les nouveaux algorithmes de remontée ne nécessitent de réaliser effectivement le changement de variable $y_i = x_i - a_i$, mais il peut etre interessant de le réaliser dans certains cas et nous indiquons comment le faire , une fois les P_i connus.

Algorithme de changement de variable.

(1) De chaque P_i prendre la partie de degré i exactement en

$x_1,..,x_m$

(2) Substituer y_1 à x_1 dans ce résultat , on obtient $R_i(y_1,...,y_m)$ qui est égal à $P_i(x_1,...,x_m)$

fin

Preuve. P_i est un polynome homogène de degré i en $y_1,...,y_m$,or quand on fait le changement de variable $y_i = x_i - a_i$, le seul terme qui peut donner un polynome de degré i est celui indiqué par l'algorithme , tous les autres sont de degré inférieur donc disparaissent.

Exemple : les résultats sur l'exemple en 6.1 donnent

$$P_2 = x_1^2 + x_1x_2 + 3x_1 + x_2 + 1$$

$$P_3 = x_1^3 + 3x_1^2 + 3x_1 + 1$$

Le terme de degré 2 de P_2 est $x_1^2 + x_1x_2$ donc $P_2 = y_1^2 + y_1y_2$.
Le terme de degré 3 de P_3 est x_1^3 donc $P_3 = y_1^3$

6.3 Complexité

La complexité moyenne de l'algorithme de factorisation est mal connue. On sait seulement qu'elle peut etre exponentielle dans le pire des cas. La complexité des étapes classiques de remontée est polynomiale en le nombre de facteurs obtenus et les degrés des polynomes .

La complexité du nouvel algorithme de remontée apparait polynomiale en le degré global p du polynome et le nombre des facteurs obtenus.

On se contentera ici d'indiquer rapidement l'étude de la complexité de l'algorithme de changement de variable.

Les calculs des R_i se font en $O(p\ Log(p))$, p étant le degré global de P en $x_1,\ldots,x_m$.

Le calcul de G_j demande m dérivations et m multiplications de polynomes. Il est meme possible de simplifier ces calculs en calculant directement les coefficients de G_j à partir de ceux de G_{j-1}.

Il en résulte que l'algorithme de calcul des Pi et par suite l'algorithme de remonté des facteurs proposé est polynomial en le nombre de facteurs sur Z[x], le degré du polynome en x et le degré global du polynome en $x_1,\ldots,x_m$.

7 - COMPARAISONS

7.1 Préliminaires.

Les comparaisons ont porté sur la forme série linéaire de l'algorithme classique , implémentée dans SAC2 et sur la forme série linéaire du nouvel algorithme.

La prédétermination des coefficients de tete n'a été utilisée dans aucun des cas , d'autre part comme son importance pour la remontée est prouvée (WA 79) , seuls des polynomes unitaires ont été utilisé , afin de ne pas compliquer trop le problème. En effet il s'agit de savoir si le nouvel algorithme peut rivaliser avec la méthode classique indépendament des résultats déjà connus sur le coefficient de tete ou les facteurs parasites. De plus une version du nouvel algorithme possédant cette caractéristique existe , ce n'est donc pas une restriction.

La question de la représentation des polynomes s'est posée lors de l'implémentation. La forme choisie dans ALDES/SAC2 et dans les autres systèmes de calcul formel est la forme récursive : Un polynome est considéré comme un polynome enune varible principale , dont les coefficients sont eux-meme des polynomes en les autres variables. Chaque coefficient est représenté comme polynome en une seconde variable principale dont les coefficients sont des polynomes en les variables restantes, etc...

exemple : $P(x,y,z) = x^2(y^2(z+2) + y(z+1) + z^3) + x(y(z^3+1)+1) + (y^2(z^3+2z) + y(z^3+2) + (z+2))$

Cette représentation convient très mal au nouvel algorithme de remontée qui remonte toutes les variables simultanément sans les différencier , en particulier l'algorithme de calculdes parties homogènes de degré i est très défavorisé puisqu'une dérivation partielle DP/Dx_i impose de descendre i fois pour chaque monome. Par contre une telle représentation s'adapte bien à la forme classique de remontée qui est récursive et au changement de variable par la règle de Horner.

La représentation la mieux adaptée serait une représentation récursive où chaque polynome serait représenté par la liste de ses monomes , un monomes étant représenté par une liste comprenant le coefficient du monome et les puissances des variables.

exemple : $3x^2yz + 4xy^2z + 4xyz - 3y^2z$ s'écrirait

$$(3,(2,1,1)),(4,(1,2,1)),(4,(1,1,1)),(-3,(0,2,1))$$

L'utilisation de cette forme permettrait un gain très important pour le calcul des termes $\sum (x_i - a_i) DP/Dx_i$.

7.2 Résultats

Nous donnons quelques exemples simples qui illustrent cependant bien les avantages et les problèmes du nouvel algorithme.

T est le temps du choix des a_i et de la factorisation sur Z.

t1 est le temps du changement de variable classique par la méthode de Horner.

t2 est le temps du calcul des parties homogènes par le nouvel algorithme.

t3 est le temps de remontée par la méthode classique.

t4 est le temps de remontée des facteurs par le nouvel algorithme. On indique de plus dans le temps de remontée consacré au calcul des décompositions en éléments simples sur Z [x] qui représente la partie d'initialisation du nouvel algorithme.

Tous les temps sont donnés en millisecondes.

Le calculs ont été fait avec ALDES/SAC2 sur un CII DPS 68 fonctionnant sous MULTICS.

Exemple 1 : Polynome à 2 variables.

$$P(x,y) = (x - y)(x + y)$$

T = 3675ms

t1 = 48ms le changement de variable étant y → y-1

t2 = 358ms

t3 = 1117 ms

t4 = 444ms dont 172ms pour la décomposition en éléments simples.

Exemple 2 : Polynome à 3 variables

$$P(x,y,z) = (x - y)(x - z)$$

T = 4043ms

t1 = 50ms pour le changement de variable y,z -> y-1,z

t2 = 556 ms

t3 = 1654ms

t4 = 508ms dont 185ms pour la décomposition

Exemple 3 : Polynome à 3 variables

$$P(x,y,z) = (x - y - z)(x + y + z)$$

T = 3946ms

t1 = 65ms pour le changement de variables y,z -> y-1,z

t2 = 814ms

t3 = 3948ms

t4 = 727ms dont 214ms pour la décomposition

Exemple 4 : Polynome à 3 variables

$$P(x,y,z) = (x^3 + xyz + 2)(x^2 + z^2 + 1)$$

T = 10078ms

t1 = 106ms pour le changement de variable y,z -> y,z-1

t2 = 1689ms

t3 = 20028ms

t4 = 5084ms

Remarques - Le temps consacré à la remontée croit rapidement en

fonction du nombre de variable et de la complexité du polynome et devient l'étape la plus couteuse.

- Le nouvel algorithme proposé pour le changement de variable se révèle peu efficace comparé à la méthode de Horner. Cependant ce résultat n'est pas significatif à cause de la représentation des polynomes utilisée.

- Le nouvel algorithme de remontée est très bon puisqu'il est toujours supérieur à la méthode classique meme après addition du temps de l'algorithme de changement de variable.

7.3 Conclusion.

Les résultats obtenus permettent d'espérer une nette amélioration de la factorisation des polynomes à plusieures variables et ils posent le problème de la représentation des polynomes dans les systèmes de calcul formel.

Il est certain que la forme récursive est est très bien adaptée pour des langages comme LISP qui supporte MACSYMA ou REDUCE par exemple. Les algorithmes utilisés sont surtout de type récursif d'où leur adéquation aux systèmes existants.

Mais il n'est pas impossible que devant les améliorations que peuvent apporter de nouveaux algorithmes comme celui proposé ici , on retienne d'autres formes. Ce serait alors la revanche de l'itération sur la récursion.

CONCLUSION

Ce travail donne les idées importantes dans le domaine de la factorisation des polynomes et propose de nouveaux algorithmes. Mais seul le cas des polynomes à coefficients entiers est abordé et il faut savoir que les algorithmes peuvent s'étendre au cas des polynomes à coefficients dans un corps de nombres algébriques voir (ME RO 76) , (LE 32).

De meme le cas des polynomes à plusieures variables est à l'ordre du jour après avoir été un peu négligé et est un domaine prometteur de nouvelles réalisations.

Cela permettra sans doute de reposer la question de la représentations des polynomes et du choix d'algorithmes adaptés à telle ou telle forme . C'est une question que les concepteurs de nouveaux système devront résoudre autrement qu'en utilisant des recettes éprouvées. De façon plus générale , c'est la possibilité pour l'utilistaeur de pouvoir modifier son système qui est en jeu.

La possibilité de généralisation du lemme de Hensel à des anneaux plus généraux n'a pas été traitée ici. Elle est néanmoins possible et cela a été étudié ,notament par Lauer dans plusieurs articles , le plus récent étant (LA 83). De meme le champ d'utilisation des méthodes p-adiques est très vaste et il n'est pas question d'en faire une liste exhaustive . Il est bon de savoir que c'est en partie grace à elles que l'un des problèmes majeurs du calcul formel dans les années 1970 , celui de la croissance des calculs intermédiaires , a été résolu.

Le cas particulier des tests d'irréductibilité de polynome n'a été que très rapidement vu , lors de la factorisation modulaire au chapitre 1. Il s'agit d'un sujet de recherche actuel sur lequel il existe une vaste littérature et qui pourrait faire l'objet d'un travail approfondi à lui seul.

Le cas des polynomes creux n'est pas non plus étudié ici. Ce cas particulier , très impportant en pratique fait aussi l'objet de nombreux travaux , notament par Zippel dans sa thèse. On peut se reporter aussi à l'article de Davenport pour certains cas particuliers (DA 83).

Enfin la nouvelle méthode peut etre utilisée pour le calcul des PGCD de polynomes à plusieures variables car le lien entre ces calculs et la factorisation sont très forts dans ce cas . Voir (MO YU 73), (WA 81) entre autres.

Il est à souhaiter que les progrès techniques et théoriques permettent de repousser plus loin les possibilités de la factorisations , limite qui est actuellement de l'ordre des polynomes de degré 100 pour le cas une variable , et beaucoup plus basse pour les polynomes à plusieurs variables.

BIBLIOGRAPHIE

(AB CA PR) S.K.ABDALI B.F. CAVINESS A. PRIDOR : "Modular Polynomial Arithmetic in Partial Fraction Decomposition" MACSYMA 1977 p253-261

(AH HO UL) A.V AHO J.E. HOPCROFT J.D. ULLMAN : "The Design and Analysis of Computer Algorithm" Addison Wesley 1974

(BE 68) E.R BERLEKAMP : Algebraic coding theory , MAC GRAW HILL ED (1968)

(BE 70) E.R. BERLEKAMP : "Factoring Polynomials over large finite fields" , Math of Comp. 24 p713-735

(B C L 82) B.BUCHBERGER G.E.COLLINS R.LOOS : COMPUTER ALGEBRA , SPRINGER VERLAG (1982)

(CA LO 82) J.CALMET R RUDIGER LOOS : "Deterministic versus Probabilistic Facorization of Integral Polynomials" EUROCAM 82

(CA 81) P. CAMION : "A deterministic Algorithm for Factoring Polynomials of $F_q[x]$ " , Marseille "COMBINATOIRE 1981"

(CA 83) P. CAMION : "Improving an Algorithm for factoring Polynomial over a finite field and constructing Large Irreducible Polynomials" , I.I.E.E. Transaction on Information theory 29 , May (1983)

(CA ZA 81) D.G. CANTOR ,H. ZASSENHAUSS : "On Algorithms For Factoring Polynomials over Finite Fields" , Math. of Comp. 36 p587-592

(CO 79) G.E. COLLINS : "Factoring Univariate Polynomials in Average Polynomial time" , EUROSAM 1979

(DA 83) J.H. DAVENPORT : "Factorization of sparse polynomials" EUROCAL 83

(DA 83) J.H. DAVENPORT : "Intégration Formelle" , Rapport de Recherche IMAG n°375 ,(1983)

(GE 60) A.O. GELFOND : Transcendantal and algebraic number , NEW YORK DOVER 1960

(HO 71) E. HOROWITZ : "Algorithms for Partial Fraction Decomposition and rational Function integration" , SYMSAM 1971 p441-457

(KA 82) E. KALTOFEN : "Factorization of Polynomials" , COMPUTER ALGEBRA , BUCHBERGER & All Editors (1982)

(KN 81) D.E. KNUTH : The Art of Computer Progamming , Vol2 "Seminumerical Algorithms" Reading , Addison Wesley 1981

(KU TO 77) H.T. KUNG D.M. TONG : "Fast Algorithms For Partial Fraction Decomposition" , SIAM J. of Algorithms 6 p582-593 (1977)

(LA 62) S. LANG : Diophantine Geometry , NEW YORK : Intersciences 1962

(LA 83) D. LAZARD : "On Polynomial Factorization" EUROCAM 1982

(LA 83) M. LAUER : "Generalised p-adic Constructions" , SIAM J. of Comp. 12 ,p395_410

(LE 82) A.K. LENSTRA : "Lattices and Factorization of Polynomials" , EUROCAM 1982

(LE 82) A.K. LENSTRA : "Factoring Polynomials over Algebraic number Fields" , Inter report IW 213/83 Math. Centrum AMSTERDAM 1983

(LE 83) A.K LENSTRA : "communication privée Londres 1983"

(LO 83) R. LOOS : "Computing rational zero of integral polynomials by p-adic expansion" , SIAM J. of Comp. 12 p286-293 (1983)

(LU 83) D. LUGIEZ : "Fast Hensel's lifting Implementation " présenté au colloque A.A.E.C.C. TOULOUSE 1983 à paraitre dans "Annals of discrete Mathematics"

(LU 84) D. LUGIEZ : "A New Lifting Process for multivariate Polynomial Factorization " à paraitre dans SIGSAM Bull.

(MI 74) M. MIGNOTTE : "An inequality about factors of Polynomials" Math. of Comp. 28 p1153-1157 (1974)

(MI 80) M. MIGNOTTE : "Factorization of univariate Polynomials A statistical study " SIGSAM Bull. 14 n°4 p41-44 (1980)

(MI 80) M. MIGNOTTE : "Test for Polynomials" , SIGSAM Bull. 14 N°1 P21-29 (1980)

(MO 77) R.T. MOENCK : "On The Efficiency of algorithms for Polynomial Factoring" , Math. of Comp. 31 p235-250 (1977)

(MO YU 73) J.MOSES D.Y. YUN : "The EZGCD Algorithm" ACM national Conference 1973 p159-166

(MU 71) D.R. MUSSER : "Algorithms For Polynomial Factorization" , Ph. D. Thesis , University of Wisconsin (1971)

(MU 76) D.R. MUSSER : "Multivariate Polynomial Factorization" J.ACM 22 p291-308 (1976)

(MU 78) D.R. MUSSER : "On the Efficiency of a Polynomial irreducibility test" J.ACM 22 p271-282 (1978)

(OS 75) A.M. OSTROWSKI : "On multiplication and factorisation of polynomials" , Aequationes Math. 13 p201-228 (1975)

(OS 76) A.M. OSTROWSKI : "On multiplication and factorization of Polynomials 2" , Aequationes Math. 14 p1-32 (1976)

(RA 80) M.O. RABIN :"Probabilistic Algorithms in Finite Fields" , SIAM J. of Comp. 9 p273-280 (1980)

(VDV 53) VAN DER WARDEN : MODERN ALGEBRA 1 , NEW YORK : UNGAR PUBL. 1953

(VI 80) G. VIRY : "Factorisation des polynomes à plusieures variables" RAIRO Informatique Théorique 14 p209-223 (1980)

(VI 82) G.VIRY : "Polynomial's Factorization over the integers" article non publié

(WAE 83) M. WAEZI : "Algorithmes de Factorisatin de Polynomes" Thèse de 3° Cycle , Poitiers 1983

(WA RO 75) P.S. WANG L.P. ROTSCHILD : "Factoring Multivariate Polynomials over the Integers" Math. of Comp. 29 p935-950 (1975)

(WA TR 79) P.S. WANG B.M. TRAGER : "New Algorithms for Polynomial Squarefree Decomposition over the Integers" SIAM J.of Comp. 8 p300-305 (1979)

(WA 76) P.S. WANG :"Factoring Multivariate Polynomials over Algebraic number Fields" Math. of Comp. 30 p324-336 (1976)

(WA 77) P.S. WANG :"Preserving Sparseness In Multivariate Polynomial Factorization" MACSYMA 1977 p55-61

(WA 78) P.S. WANG :"An Improved Multivariate Polynomial Factoring Algorithm" Math. of Comp. 32 p1215-1231 (1978)

(WA 79) P.S. WANG : "Parallel p-adic Construction in The Univariate Polynomial Factoring Algorithm " MACSYMA 1979 p310-318

(WA 800 P.S. WANG : "THE EZ-GCD Algorithm" SIGSAM Bull. 14 n 2 p50-60 (1980)

(WA 83) P.S. WANG : " The Early Detection of True Factors in Univariate Polynomial Factorization" EUROCAL 1983 p225-235

(WE RO 76) P.J. WEINBERGER L.P. ROTSCHILD : "Factoring Polynomial over Algebraic Number Field" ACM Tans. Math. Software 2 p335-350 (1976)

(YU 76) D.Y. YUN :" On Squarefree Decomposition Algorithm" SYMSAC 1976 p26-35

(YU 77) D.Y. YUN : "On The Equivalence of Polynomial GCD and Squarefree Factorization Problems" MACSYMA 1977 p65-70

(ZA 69) H. ZASSENHAUSS : "On Hensel Factorization 1 " J. Of Number Theory 1 p291-311 (1969)

(ZA 78) H. ZASSENHAUSS : "A Remark on The Hensel Factorization Method" Math. of Comp. 32 p287-292 (1978)

(ZA 81) H. ZASSENHAUS : "Polynomial time Factoring of Integral Polynomials" SIGSAM Bull. 15 p6-7 (1981)

(ZI 79) R.E. ZIPPEL : "Probabilistic Algorithms for Sparse Polynomials" EUROSAM 1979

(ZI 79) R.E. ZIPPEL : "Probabilistic Algorithms for Sparse Polynomials" Ph.D. Thesis M.I.T. 1979

iuphlp.aldes

```
        ls=iuphlp(p,a,l,cc)
$(calcul des facteurs du polynome a a partir des facteurs modulo p
contenus dans la liste l par le lemme de hensel parallele)
  (1)  $(initialization)
       ls=( ). la=l. lc=( ). q=p. r=length(l). ap=a.
       m=iupbcf(p,ap). iroot(m,r.ml,tl). lb=iuphli(p,ap,la).
  (2)  $(iteration)
       while q lt m do ( la=iuphlf(q,m,ap,la,lb). qq=imin(iexp(q,2),m).
       if qq ge m then go to 3. if qq ge ml then (
       iuptvf(qq,ap,la.sl,lz,b). ls=conc(sl,ls). if lz eq ( ) then
       return. if sl ne ( ) then (ml=iupbcf(p,b). if qq ge ml
       then la=ipflc(l,qq,( ),b,lz,cc) else la=iuphlp(p,b,lz,cc).
       ls=conc(la,ls). return)). lb=iuphlc(q,a,la,lb). q=qq).
  (3)  $(vrais facteurs)
       lc=ipflc(l,m,( ),a,la,cc). ls=comp(ls,lc)..
..
```

iuphli.aldes

```
        lb=iuphli(p,a,l)
  (1)  $(initialization)
       lb=( ). la=l. al=pldcf(a). w=list2(0,1). q=p. qq=iexp(q,2).
  (2)  $(calcul de as)
       while la ne ( ) do ( adv(la.ri,la). w=mippr(l,qq,w,ri)). la=l.
       as=ipdif(l,a,ipip(l,al,w)). as=miphom(l,q,ipiq(l,as,q)).
  (3)  $(calcul des ai)
       while la ne ( ) do ( adv(la.ri,la). wi=ipq(l,w,ri).
       wi=miphom(l,q,wi). miupqr(q,wi,ri.qi,di).
       mupegc(q,ri,di.c,ei,ai). lb=comp(ai,lb)). lb=comp(as,inv(lb))..
..
```

iuphlf.aldes

```
        ls=iuphlf(q,m,a,l,lb)
  (1)  $(initialization)
       ls=( ). la=l. adv(lb.as,lba). qq=imin(iexp(q,2),m).
  (2)  $(rendre as unitaire)
       al=miinv(q,pldcf(a)). as=miphom(l,q,ipip(l,al,as)).
  (3)  $(new factors)
       while la ne ( ) do ( adv(la.ri,la). adv(lba.ai,lba).
       ap=mippr(l,qq,ai,as). miupqr(q,ap,ri.qi,rsi).
       ri=ipsum(l,ri,ipip(l,q,rsi)). ri=miphom(l,qq,ri).
       ls=comp(ri,ls)). ls=inv(ls)..
..
```

iuphlc.aldes

```
        lbs=iuphlc(q,a,l,lb)
  (1)  $(initialization)
       lbs=( ). la=l. lba=red(lb). i=list2(0,1). w=i.
       qq=iexp(q,2). al=pldcf(a). qp=iexp(qq,2). t=list2(0,0).
  (2)  $(new as)
       while la ne ( ) do (adv(la.ri,la). w=mippr(l,qp,w,ri)).la=l.
       as=ipdif(l,a,ipip(l,al,w)). as=miphom(l,qq,ipiq(l,as,qq)).
  (3)  $(new coefficients)
       while la ne ( ) do ( adv(la.ri,la). adv(lba.ai,lba).
       wi=ipq(l,w,ri). ti=ipprod(l,ai,wi). t=ipsum(l,t,ti)).
       t=ipdif(l,t,i). t=ipiq(l,ipneg(l,t),q). la=l. lba=red(lb).
       while la ne ( ) do (adv(la.ri,la). adv(lba.ai,lba).
       ti=ipprod(l,ai,t). ipqr(l,ti,ri.qi,asi).
```

iupvls.aldes

```
        ls=iupvls(p,a,l,cc)
$(lemme de hensel serie)
  (1)  $(initialisation)
       ap=a. c=a. lc=( ). ls=( ). m=iupbcf(p,a). r=length(l).
       la=l. iroot(m,r.m1,t1).
  (2)  $(remontee des facteurs)
       while red(la) ne ( ) do ( adv(la.ab,la).
       iupvsf(p,m,m1,ap,c,ab.as,bs,cs). if cs ne ( ) then (c=cs.
       ls=comp(as,ls). ap=cs. m=iupbcf(p,cs)) else ( lc=comp(as,lc).
       c=bs)). miupqr(m,bs,list2(0,pldcf(bs)).bs,r). lc=comp(bs,lc).
  (3)  $(facteurs)
       lz=ipflc(1,m,( ),ap,lc,cc). ls=conc(ls,lz)..
..
```

iupvsf.aldes

```
       iupvsf(p,m,m1,ap,c,ab.a,b,cs)
  (1)  $(initialisation)
       ql=p. qls=iprod(ql,ql). cp=miphom(1,qls,c). a=ab.
       miupqr(qls,cp,a.b,r). b=miphom(1,ql,b). mupegc(p,a,b.d,s,t).
       cs=( ). i=list2(0,1).
       if ql eq m then return.
  (2)  $(compute y,z)
       r=miphom(1,ql,ipiq(1,r,ql)). qls=imin(m,iprod(ql,ql)).
       xa=mippr(1,ql,t,r). miupqr(ql,xa,a.q1,x).
       as=miphom(1,qls,ipsum(1,a,ipip(1,ql,x))).
       qs=imin(iprod(qls,qls),m). cp=miphom(1,qs,c).
       miupqr(qs,cp,as.bb,r). bb=miphom(1,qls,bb).
       if qls ge m1 then (tesfac(qls,ap,as.ak,cs). if cs ne ( )
       then (a=ak. return )).
       if qls ge m then (a=as. b=bb. return).
  (4)  $(compute y1,z1)
       ra=ipprod(1,as,s). rp=ipprod(1,bb,t). ra=ipsum(1,rp,ra).
       ra=ipdif(1,ra,i). u1=ipiq(1,ra,ql). miupse(ql,a,b,s,t,u1.y1,z1).
  (5)  $(compute ss,ts)
       ra=ipip(1,ql,y1). ss=mipdif(1,qls,s,ra). ra=ipip(1,ql,z1).
       ts=mipdif(1,qls,t,ra).
  (6)  $(advance)
       ql=qls. a=as. b=bb. s=ss. t=ts. go to 2..
..
```

iupvls.aldes

```
        ls=iupvls(p,a,l,cc)
$(lemme de hensel serie)
  (1)  $(initialisation)
       ap=a. c=a. lc=( ). ls=( ). m=iupbcf(p,a). r=length(l).
       la=l. iroot(m,r.m1,t1).
  (2)  $(remontee des facteurs)
       while red(la) ne ( ) do ( adv(la.ab,la).
       iupvsf(p,m,m1,ap,c,ab.as,bs,cs). if cs ne ( ) then (c=cs.
       ls=comp(as,ls). ap=cs. m=iupbcf(p,cs)) else ( lc=comp(as,lc).
       c=bs)). miupqr(m,bs,list2(0,pldcf(bs)).bs,r). lc=comp(bs,lc).
  (3)  $(facteurs)
       lz=ipflc(1,m,( ),ap,lc,cc). ls=conc(ls,lz)..
..
```

iupvsf.aldes

```
      iupvsf(p,m,m1,ap,c,ab.a,b,cs)
  (1)  $(initialisation)
       ql=p. qls=iprod(ql,ql). cp=miphom(1,qls,c). a=ab.
       miupqr(qls,cp,a.b,r). b=miphom(1,ql,b). mupegc(p,a,b.d,s,t).
       cs=( ). i=list2(0,1).
       if ql eq m then return.
  (2)  $(compute y,z)
       r=miphom(1,ql,ipiq(1,r,ql)). qls=imin(m,iprod(ql,ql)).
       xa=mippr(1,ql,t,r). miupqr(ql,xa,a.ql,x).
       as=miphom(1,qls,ipsum(1,a,ipip(1,ql,x))).
       qs=imin(iprod(qls,qls),m). cp=miphom(1,qs,c).
       miupqr(qs,cp,as.bb,r). bb=miphom(1,qls,bb).
       if qls ge m1 then (tesfac(qls,ap,as.ak,cs). if cs ne ( )
       then (a=ak. return )).
       if qls ge m then (a=as. b=bb. return).
  (4)  $(compute y1,z1)
       ra=ipprod(1,as,s). rp=ipprod(1,bb,t). ra=ipsum(1,rp,ra).
       ra=ipdif(1,ra,i). ui=ipiq(1,ra,ql). miupse(ql,a,b,s,t,ui.y1,z1).
  (5)  $(compute ss,ts)
       ra=ipip(1,ql,y1). ss=mipdif(1,qls,s,ra). ra=ipip(1,ql,z1).
       ts=mipdif(1,qls,t,ra).
  (6)  $(advance)
       ql=qls. a=as. b=bb. s=ss. t=ts. go to 2..
..
```

iuphlv.aldes

```
        ls=iuphlv(p,a,l,c)
$(calcul des facteurs du polynome a a partir des facteurs modulo p
contenus dans la liste l)
  (1)  $(initialisation)
       ls=( ). lc=( ). ap=a. la=l. m=iupbcf(p,ap). r=length(la). q=p.
       iroot(m,r.m1,t1). iuphvi(p,ap,la.lb,as).
  (2)  $(new factor) iuphvf(q,m,ap,as,la,lb.la,as). qq=iprod(q,q).
  (3)  $(test de fin) if qq ge m then (lc=ipflc(1,qq,( ),ap,la,c).
       ls=comp(ls,lc). return ).
  (4)  $(detection de facteurs)
       if qq ge m1 then ( iuptvf(qq,ap,la.s1,lz,b).
       if lz eq ( ) then (ls=conc(s1,ls). return).
       if s1 ne ( ) then (ls=conc(s1,ls). m=iupbcf(p,b).
       if qq ge m then ( lz=ipflc(1,qq,( ),b,lz,c). ls=conc(lz,ls).
       return) else ( ap=b. lb=reinit(q,b,lz,la,lb). la=lz))).
  (5)  $(nouveaux coefficients)
       lb=iuphvc(q,m,a,la,lb).  q=qq. go to 2..
..
```

iuphvi.aldes

```
        iuphvi(p,a,l.lb,as)
  (1)  $(initialization)
       q=p. qq=iexp(q,2). la=red(l). lb=( ). ap=miphom(1,qq,a).
       w=list2(0,1).
  (2)  $(calcul de as)
       while la ne ( ) do ( adv(la.ri,la). w=mippr(1,qq,w,ri)).
       miupqr(qq,ap,w.r1,as). la=red(l). ap=miphom(1,q,ap).
  (3)  $(calcul de la liste lb)
       while la ne ( ) do ( adv(la.ri,la). miupqr(q,ap,ri.wi,di).
       miupqr(q,wi,ri.qi,di).  mupegc(q,di,ri.c,ai,ei).
       lb=comp(list2(ai,ei),lb)). lb=inv(lb)..
..
```

iuphvf.aldes

```
        iuphvf(q,m,a,as,l,lb.ls,ass)
$(calcul des facteurs de a modulo q**2 a partir des facteurs modulo q
  et de la difference ass de a et du produit des facteurs)
  (1)  $(initialization)
       ls=( ). lba=lb. qq=imin(iexp(q,2),m). qp=imin(iexp(qq,2),m).
       la=red(l). w=list2(0,1). ak=miphom(1,q,ipiq(1,as,q)).
       ap=miphom(1,qp,a).
  (2)  $(facteurs i g.e. 2)
       while la ne ( ) do ( adv(la.ri,la). adv(lba.ci,lba).
       ai=first(ci). li=mippr(1,q,ai,ak). miupqr(q,li,ri.qi,rsi).
       ri=ipsum(1,ri,ipip(1,q,rsi)). ls=comp(miphom(1,qq,ri),ls)).
  (3)  $(premier facteur et ass)
       la=ls. while la ne ( ) do (adv(la.ri,la). w=mippr(1,qp,ri,w)).
       miupqr(qp,ap,w.r1,ass). ls=comp(r1,inv(ls))..
..
```

iuphvc.aldes

```
        lbs=iuphvc(q,m,a,ls,lb)
$(calcul des coefficients lemme de hensel methode viry)
  (1)  $(initialization)
       la=red(ls). lbs=( ). lba=lb. i=list2(0,1).  qq=iexp(q,2).
  (2)  $(coefficients)
       w=miphpm(1,qq,a). while la ne ( ) do (adv(la.ri,la).
       adv(lba.ci,lba). miupqr(qq,w,ri.wi,di). adv2(ci.ai,ei,ci).
       miupqr(qq,wi,ri.qi,di).
       a1=ipprod(1,ri,ei). a2=ipprod(1,ai,di). a1=ipsum(1,a1,a2).
       ti=ipdif(1,i,a1). ti=miphom(1,q,ipiq(1,ti,q)).
       miupse(q,di,ri,ai,ei,ti.yi,zi). ai=ipsum(1,ai,ipip(1,q,yi)).
       ei=ipsum(1,ei,ipip(1,q,zi)). ai=miphom(1,qq,ai).
```

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU le rapport de présentation de Monsieur J. CALMET, Chargé de recherche

Monsieur Denis LUGIEZ

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 10 janvier 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,