数理解析研究所講究録
920巻 1995年 133-148

15.

$\mathbf{Z}_p$ 上の多項式の因数分解

— 高速化技法・ベクトル処理・並列処理 —

村尾 裕一 (東京大学)

15.1 はじめに

多項式の因数分解は，今や，数式処理システムには必要不可欠な機能であり，また，数式処理の様々な基本アルゴリズム中においても必要となる最も基本的な演算である．

そのような基本アルゴリズムのひとつとして，多項式の補間法による決定があげられる．多項式補間とは，未知の多項式の多数の数値群から目的とする多項式を構成する方法だが，その数値群の計算には，ベクトル処理や (超) 並列処理が適用可能な場合が多く，効率上，実際に適用することが望まれる．補間法のアルゴリズムとしては，Lagrange や Newton によるものが古くから知られているが，これら古典的な方法は，実際上は，疎な多項式に対しては使いものにならない．近年，疎な ($\mathbf{Z}$ 係数の) 多項式を決定するための効率の良いアルゴリズムが BenOr と Tiwari により発見され [2]，以降，幾つかのその発展形が発表されている: (Kaltofen & Lakshman) Toeplitz 方程式と Vandermonde 方程式の解法部分への，より効率の良い既知の算法の適用 [6]; (Kaltofen 他) 数係数膨張を抑止するための p-adic なアルゴリズムと $\mathbf{Q}$ 係数の場合への拡張 [7]; (Murao & Fujise) 数値群のデータ並列による計算を念頭においた，複数の法 (大きな素数) の利用 [12]. いずれのアルゴリズムにおいても，一変数多項式の因数分解 (前二者では $\mathbf{Z}$ 上，後二者では $\mathbf{Z}_p$ 上) が必要となる．この一変数多項式は，$\mathbf{Z}$ 或は $\mathbf{Z}_p$ 上で完全に線形因子にまで分解されることは既知であり，また，その次数は，補間により決定する多項式中の単項式の個数に等しい．よって，行列式の展開等で巨大な多項式を求める場合には，因数分解する多項式はかなり高次となる．実際，上記の我々のアルゴリズムの開発・実用化において，この因数分解がネックとなる (ことが REDUCE/Rlisp による実験で明かとなっている) [12].

筆者は，この困難を実際・実用上いかに克服しうるか又すべきかを明かにするために，問題をより一般化し，表題のとおりの因数分解について，様々な実験を行っている．即ち，各種のアルゴリズムについて,「インプリメント上どのようなことが問題となるのか?」,「実行性能はどのアルゴリズムが優れ，また，どこまで得られるのか?」，[13] にも指摘があるとおり「Berlekamp の因数分解アルゴリズムは，基本的に行列の処理ゆえベクトル処理が可能だが，どの程度の効果があるのか?」，さらに一般に「ベクトル処理や並列処理を如何に用いるのか? また，その効果は?」等の知見をうることが，その目的である．

本稿は，この計算機実験の，主にベクトル処理についての，現時点までの結果報告である．以降では，先ず，既存のアルゴリズムとその complexity 等のまとめを行い，その後，ベクトル処理の仕方等について論じ，その効果を計測結果と共に示す．その実験結果についての簡単な考察を行った後，最後に，並列処理についても言及する．並列処理の具体的方法や結果については，[12] や [18] を参照されたい．

15.2 アルゴリズムのまとめ

$\mathbf{Z}_p$ 上の無平方 (square-free) で monic な多項式 $U(x)$ を因数分解する．但し，$n = \deg(U)$, r を既約因子の個数とする．勿論 r は未知である．$x^{pk} \mod U(x)$ の x^i の係数を $q_{ik} \in \mathbf{Z}_p$ とおき，$n \times n$ 行列 Q を以下のとおり定義する:

$$Q = \begin{pmatrix} q_{00} & q_{01} & \cdots & q_{0\,n-1} \\ \vdots & \vdots & & \vdots \\ q_{n-1\,0} & q_{n-1\,1} & \cdots & q_{n-1\,n-1} \end{pmatrix}.$$

■ Berlekamp のアルゴリズム [3]

(1) $Q - I$ の零空間を求める．但し，I は $n \times n$ の単位行列．より具体的には，行列 $Q - I$ の列の消去を行い:

(a) $\mathrm{rank}(Q - I) = n - r$ を求め，因子の個数 r を決定する．

(b) $Q\vec{v}_j = \vec{v}_j$ を満たすベクトル $\vec{v}_j = (v_{j0}, v_{j1}, \ldots, v_{jn-1})^T$ を求め，$V_j(x) = \sum_{i=0}^{n-1} v_{ji} x^i$ とする．ここで，$V_j(x)$ が次の合同式を満たすことに注意:

$$V_j(x)^p \equiv V_j(x) \mod U(x).$$

(2) r 個の因子が求まるまで，各 $V_j(x)$ に対し，$\alpha = 0, 1, \ldots, p-1$ に対し $V_j(x) - \alpha$ と $U(x)$ またはその因子との GCD 計算を繰り返す．

■ Zassenhaus — 有用な α の決定 [17]

上記 Berlekamp のアルゴリズムでは，最悪の場合，GCD 計算を全ての α に対して繰り返すことになるが，$p \gg 1$ の場合，その繰り返しの回数が問題となるばかりでなく，殆んどの GCD 計算が無駄な (因子の得られない) 計算である．しかし，いずれの $v(x) = V_j(x)$ についても，

- $v(x)^k \bmod U(x), k = 0, 1, ..., r$ は線形従属ゆえ，
- $w(v(x)) \equiv 0 \bmod U(x)$ を満たす多項式 $w(X)$ が存在する．最低次の $w(X)$ を求めれば，
- $w(X) = 0$ の解 $\alpha \in \mathbf{Z}_p$ が GCD 計算で non-trivial な因子を与える．

より具体的には

(1) 以下の行列消去を行い零空間ベクトル $\vec{w} = (w_0, w_1, ...)^T$ を求め，$w(X) = \sum_k w_k X^k$ とおく．

- **(a)** $v(x)^k \bmod U(x)$ の係数を第 k 列の列ベクトルとする行列を順次構成し，
- **(b)** その第 k 列を $0 \sim (k-1)$ 列により消去する．
- **(c)** 線形従属であることが判明した時点で終了．

(2) $w(X)$ の全解 $(\in \mathbf{Z}_p)$ を求める．

.....

上記の 2 つのアルゴリズムにおいて注意すべき事柄を列挙する．

- 必ずしも $\vec{v}_j$ を全て求める必要はない．
- Berlekamp のアルゴリズムで，p が大きい場合，$V_j(x)$ の j に関する繰り返しと α に対する繰り返しは，r と p の大きさにより適宜入れ換えて実行すべきである．
- $U(x)$ の既約でない因子 $u(x)$ の中には，上記の $V_j(x)$ を用いた GCD 計算では分解できないものがあることに注意する必要がある．$V_j(x) \bmod u(x) \in \mathbf{Z}_p$ の場合である．

次数別因数分解 (Distinct Degree Factorization: 以下 DDF)

$H_k(x)$ を，$U(x)$ の因子のうち次数が k である全ての既約因子の積とする．各 $H_k(x)$ は，$U^{(1)}(x) = U(x)$ とおき，$k = 1, 2, ...$ に対し (小さい順に) $U^{(k+1)}(x) = 1$ となるまで次式を計算することにより求められる．

$$H_k(x) \Leftarrow \gcd(x^{p^k} - x, U^{(k)}(x)),$$
$$U^{(k+1)}(x) \Leftarrow U^{(k)}(x)/H_k(x).$$

$d_k = \deg(H_k)$ とすると，$H_k(x)$ の既約因子の個数は d_k/k である．

以下のアルゴリズムでは，この $H_k(x)$ の因数分解を行う．各 $H_k(x)$ の既約因子の次数と個数が既知ゆえ，完全な分解が得られたか否かの判定は容易にできるが，以下でアルゴリズムが「確率的」というのは，全ての既約因子を得るまでの手続きが非決定的，ということである．

■DDF + Cantor & Zassenhaus のアルゴリズム [4] — 確率的

任意の多項式 $A(x) \in \mathbf{Z}_p[x]$ に対し，次式が成り立つことを利用する．

$$A(x^{p^k}) - A(x) \equiv A(x)^{p^k} - A(x) \equiv A(x)(C-1)(C+1) \equiv 0 \mod H_k(x),$$

但し，$C = A(x)^{(p^k-1)/2} \mod H_k(x)$. アルゴリズムとしては，$A(x)$ を random に生成し，$H_k(x)$ 又はその因子と $C-1$ との GCD 計算を繰り返す．

■DDF + Ben-Or のアルゴリズム [1] — 確率的

McEliece operator T_k は，各 $H_k(x)$ に対し，次式により定義される [9]:

$$T_k(y) = y^{p^0} + y^{p^1} + \cdots + y^{p^{k-1}}.$$

$T_k(y)^p - T_k(y) = y^{p^k} - y$ ゆえ，任意の $A \in \mathbf{Z}_p[x]$ に対し次式が成り立つ．

$$\begin{aligned} T_k(A)^p - T_k(A) &\equiv 0 \mod H_k(x) \\ &= T_k(A)\left(T_k(A)^{(p-1)/2} - 1\right)\left(T_k(A)^{(p-1)/2} + 1\right) \left(= \prod_{\alpha=0}^{p-1} (T_k(A) - \alpha) \right). \end{aligned}$$

アルゴリズムは，$H_k(x)$ またはその因子が完全に分解されるまで，以下の手順を繰り返す:

(1) $A \in \mathbf{Z}_p[x]$ を random に生成し，$T_k(A)$ を計算する．

(2) $\alpha \in \mathbf{Z}_p$ を random に生成し，$B = (T_k(A) + \alpha)^{(p-1)/2} \pm 1$ を計算する．

(3) B と $U(x)$ またはその因子との GCD を計算する．

(4) 充分に多くの α に対し，(2) 及び (3) を繰り返す．

ここで，ひとつの A によって $H_k(x)$ の因子を完全に分離しうるかは，その確率が $p \gg d_k/k$ の場合非常に高いとはいえ，あくまでも確率的であることに注意．

■DDF + Shoup のアルゴリズム [14]

(1) $g_i(x)$ を，$(Y - x^{p^0})(Y - x^{p^1}) \cdots (Y - x^{p^{k-1}})$ の Y^i の係数とする:

$$(Y - x^{p^0})(Y - x^{p^1}) \cdots (Y - x^{p^{k-1}}) = Y^k + g_{k-1}(x)Y^{k-1} + \cdots + g_0(x).$$

(2) 全ての $\alpha \in \mathbf{Z}_p$ 及び各 $g_i(x), i = 0, 1, \ldots, k-1$ に対し，$H_k(x)$ またはその因子と，$g_i(x) + \alpha$ および $(g_i(x) + \alpha)^{(p-1)/2} - 1$ との GCD 計算を，全ての因子が求まるまで繰り返す．

ここで，(2) のステップ中，α についての繰り返しと $g_i(x)$ についての繰り返しのいずれを，如何なる場合に優先させるべきかは，[15] に詳しい．

15.3 計算の複雑さや技法について

本節では，より具体的な計算法について言及すると同時に，各アルゴリズムの計算量とメモリ所要量をまとめる．後節では，実行時間の計測結果の検討を計算量に基づいて行う．また，メモリ所要量は，インプリメントの際に，実際にどれくらいのメモリが必要になるのか，メモリはどのように使い回せるのか，また，どの程度の規模まで計算できるのかを知る上で必要となる．

計算量

多項式の乗算に FFT や Karatsuba 等の技法を用いない場合の各アルゴリズムのステップ毎の計算量を，表 1 に [15] より引用する．

表 1　各アルゴリズムのステップ毎の計算量

	Berlekamp	C&Z 及び Ben-Or	Shoup
行列 Q の構成	$O(n^2p)$ または $O(n^3+n^2\log p)$	←	←
Q の消去 | DDF	$O(n^3+n\log p)$	$O(n^3+n\log p)$	←
GCD	$O(n^3+n^3p)$	$O(n^3+n^3\log p)$	$O(n^3p+n^2)$ または $O(n^3p^{1/2}\log p+n^2)$

行列 Q の構成について

表 1 中 2 つの計算量を併記してあるのは，次の算法の違いによる．

- $O(n^2p)\cdots x^k \bmod U(x)$ を，shift-add により，$k=1,2,\ldots,(n-1)p$ と順次計算していく．
- $O(n^3+n^2\log p)\cdots F$ を $U(x)$ 又は $H_k(x)$ とし，但し $n=\deg(F)$，$j=n,n+1,\ldots,2n-2$ に対し $x^j \bmod F=\sum_{i=0}^{n-1}R_{ji}x^i$ の表 (R_{ji}) を用意し，高次の冪乗の計算を 2 進法的に行う．

$p\gg 1$ の場合後者が優れるが，表 $\{R_{ji}\}$ の分だけ余分なメモリが必要となる．但し，この表は様々な部分で利用可能・必要となる．本稿のインプリメントでは，$p<n$ の場合に前者を，そうでない場合に後者を用いている．

高次の冪乗の計算について

DDF 及び DDF 後のアルゴリズムでは，多項式の高次の冪乗 $(\bmod F)$ の計算が必要となる．

- 任意の $A(x)\in \mathbf{Z}_p[x]$ について，$A(x)^p\equiv A(x^p) \bmod p$ が成り立つ．よって，$A(x)=\sum_{i=0}^{n-1}A_ix^i$ の p 乗 $A(x)^p$ の係数は，行列 Q を用いて，$Q\,(A_0,A_1,\ldots,A_{n-1})^T$ により計算される．当然，$A(x)^{p^{k+1}}$ も $A(x)^{p^{k+1}}\equiv A(x^{p^k})^p$ により，行列 Q を用いて計算する．
- Cantor & Zassenhaus のアルゴリズムで用いる $A(x)^{(p^k-1)/2}$ は，$p^k-1=(1+p+p^2+$

$\cdots + p^{k-1})(p-1)$ に注意し，次式により行列 Q を用いて計算する．

$$A(x)^{(p^k-1)/2} = \left(\prod_{j=0}^{k-1} A(x^{p^j})\right)^{(p-1)/2}.$$

但し，DDF 以後のアルゴリズムで必要となる Q-行列は，各 $H_k(x)$ に対する行列 $Q^{(k)}$ である．行列 $Q^{(k)}$ は，p や d_k が大きい場合，DDF の際に用いた，$U(x)$ に対する Q から (各列の H_k による剰余として) 計算すると良い．

線形因子の積の分解について

$L(x) \in \mathbf{Z}_p[x]$ を異なる d 個の線形因子の積 $L(x)$ とする．この分解は，次のいずれかにより行う．

(a) $x = 0, 1, \ldots, p-1$ の代入⇒ 計算量は $O(dp)$ だが，その係数は小さく，計算も容易．

(b) $\gcd(L(x), (x-\alpha)^{(p-1)/2} \pm 1)$ による二分探索⇒ 計算量は，表 1 から，$O(d^3 + d^3 \log p)$．また，$x^k \bmod L(x)$ の係数の表 (大きさは d^2) が必要．

メモリ所要量について ... 次数 n について 2 乗以上のもののみをまとめる．

Berlekamp : 総量 $2n^2$

	表 R_{ki}	行列 Q	$V_j(x)$	その他
線形因子の分離と求根	n^2			d^2
行列 Q の作成	○	n^2		
行列 $Q-I$ の消去		○	rn	
GCD 計算による分離			○	

Zassenhaus : 総量 $5/2n^2 \sim 3n^2$; DDF も行う場合は Q の保存のために $+n^2$

線形因子の分離と求根	n^2			d^2
行列 Q の作成	○	n^2		
行列 $Q-I$ の消去		○	rn	
$w(X)$	○		○	rn (V_jのべき乗)
$w(X)=0$ の求根				d^2 ($d=\deg(w_j)$)
GCD 計算による分離			○	

Cantor&Zassenhaus 及び Ben-Or : 総量 $3n^2$

	表 R	行列 Q	行列 $Q^{(k)}$
行列 Q の作成	n^2	n^2	
DDF		○	
$Q \to Q^{(k)}$		○	n^2
GCD 計算による分離	○	(保存)	○

15.4 ベクトル処理の可能性と必要性

前節までで，実際の計算において行列演算を多用することは述べたので，いずれのアルゴリズム中にも，ベクトル処理に適合する処理が多々存在することは，既に明かであろう．以下に，ベクトル処理に関する事柄を，より具体的に列挙する．

- p を一語に納まる程度の大きさに選び，$\mathbf{Z}_p$ 要素の演算を倍精度で行った後，剰余の計算を行う [11].
- 一変数多項式は密であるとし，配列 (ベクトル) で表現する．その演算はベクトル処理可能．
- 行列 Q に関する処理は，多項式の高次冪乗計算も含め，ベクトル処理が可能．
- 計算量における n の因子の指数が，実際の効果として，小さく見えるようになることが期待される．
- 但し，n が大きくなければ実効性能は上がらない．よって，n が一定である GCD 計算より前の段階まではベクトル処理の効果が上がるが，扱う多項式の次数が次第に低下する GCD 計算の部分では，上記の多項式演算のベクトル処理が可能とは言え，ベクトル処理の効果はあまり期待できないであろう．後述のとおり，GCD 計算の繰り返しは基本的に探索ゆえ，並列処理向きである．
- 線形因子のみの積の分解 (求解) では，多数の α に対する (代入が) ベクトル処理可能．

15.5 ベクトル処理の実験と計測結果

これまでに，15.2節に示したアルゴリズム (Shoup のアルゴリズム: 2 変数多項式の演算が必要) を FORTRAN 77 でインプリメントし，(ベクトル処理型) スーパーコンピュータ HITAC S-3800 : HI-OSF/1-MJ (@東京大学大型計算機センター) 上で実験と計測を行っている．ソースの量は，注釈行を含め，共通して用いる副プログラム群が 2000 行，各アルゴリズムのドライバーが 500 行程度である．S-3800 の処理性能は，ベクトル演算器は 8Gflop/s の性能を有するが，スカラー演算器のみを使用する通常の数式処理システムの実行では SparcStation 2 の数倍程度である．

- 現時点で， Berlekamp, Zassenhaus 及び DDF で得られた各因子に対する Zassenhaus($Q^{(k)}$ は Q より計算), Cantor&Zassenhaus, Ben-Or ($Q^{(k)}$ は作成し直す) アルゴリズムを実現済み．
- Berlekamp と Zassenhaus のアルゴリズムでは，Q-行列の作成により得られる $x^p \bmod U(x)$ を用いて，線形因子だけを最初に除去．
- 線形因子の分離は，p の大きさによって代入または二分探索を行う．実験結果より，$p > 20000$ の場合に二分探索とした．
- 実行用のベクトル化プログラムは，各ループの実行直前にベクトル長が充分に大きいかを検査し，小さい場合にはスカラー処理を行うようにコンパイルした (そのオーバーヘッドを予想し改良も試みたが，結果的に無視しうる程度であった)．

いくつかの例題による実行時間を示す．時間の単位は，特に示さない限りマイクロ秒である．但し，表中に記載のないアルゴリズムは，長大な実行時間を要したために計測を中断した場合である．

(1) [15] 中の問題:

$$\text{(表 2)}\ f_1(x) = x^8 + x^6 + 10x^4 + 10x^3 + 8x^2 + 2x + 8$$

$$\begin{aligned}\text{(表 3)}\ f_2(x) = {} & x^{18} + 9x^{17} + 45x^{16} + 126x^{15} + 180x^{14} + 27x^{13} - 540x^{12} - 1215x^{11} \\ & +1377x^{10} + 15444x^9 + 46899x^8 + 90153x^7 + 133893x^6 + 125388x^5 \\ & +29160x^4 - 32076x^3 + 26244x^2 - 8748x + 2916\end{aligned}$$

(2) 次数が比較的高い場合 ($n = 100$) $\cdots$ $p = 43, 1979, 227951$ に対し，random に生成した $\mathbf{Z}_p$ 上の 100 次の多項式の因数分解．

- $p = 43$: 既約因子の個数は，1, 2, 3, 4, 5, 7, 9, 10 次が各 5, 7, 2, 10, 2, 1, 1, 1 ずつ．

step	Berlekamp	Zassenhaus	DDF+Zass	BenOr
	SPU+ VPU	SPU+ VPU	SPU+ VPU	SPU+ VPU
linear facs	1064+ 598	1063+ 592	44+ 6	
Q matrix	4365+ 3356	4299+ 3273	4766+ 3631	5054+ 3921
null sp|DDF	8856+ 6515	8816+ 6511	4748+ 2621	4782+ 2705
GCD	165868+100580	102265+62232	16549+ 9945	21405+14337
total	180153+111049	116443+72608	26063+16237	31241+20963

- $p = 1979$: 既約因子の個数は，1, 2, 3, 4, 6, 8, 9 次が各 6, 13, 3, 3, 2, 1, 3 ずつ．

step	Berlekamp	Zassenhaus	DDF+Zass	BenOr
	SPU+ VPU	SPU+ VPU	SPU+ VPU	SPU+ VPU
linear facs	2523+ 1443	2497+ 1432	223+ 35	
Q matrix	13731+ 11435	13648+11383	15521+13003	15573+13068
null sp|DDF	8784+ 6455	8737+ 6455	5796+ 2976	5789+ 2974
GCD	909639+417412	17290+ 8985	20439+12084	32424+21584
total	934677+436745	42172+28255	41756+28063	53784+37626

- $p = 227951$: 既約因子の個数は，1, 2, 3, 5, 6, 8, 10 次が各 5, 8, 3, 4, 2, 1, 3 ずつ．

step	Zassenhaus	DDF+Zass	BenOr
	SPU+ VPU	SPU+ VPU	SPU+ VPU
linear facs	6851+ 4321	2761+ 1507	
Q matrix	15489+12899	17406+14599	17334+ 14419
null sp|DDF	9331+ 6771	7235+ 3410	7223+ 3404
GCD	26185+14351	29356+17067	1130903+724477
total	57856+38342	53997+35076	1155460+742300

表 2　$f_1(x)$ の因数分解の計算時間 (単位はマイクロ秒)

p	因子の次数	step	Berlekamp	Zassenhaus	DDF+Zass	C&Z	BenOr
			SPU+VPU	SPU+VPU	SPU+VPU	SPU+VPU	SPU+VPU
5	1, 3, 4	linear facs	105+ 28	106+ 28			
		Q matrix	83+ 33	87+ 35	116+ 52	115+ 51	116+ 51
		null sp|DDF	74+ 28	74+ 28	186+ 63	185+ 64	287+ 65
		GCD	118+ 40	138+ 41	28+ 1	27+ 1	28+ 1
		total	380+ 129	405+ 132	330+ 116	327+ 116	431+ 117
13	1, 3, 4	linear facs	134+ 41	134+ 41			
		Q matrix	164+ 76	166+ 76	196+ 99	195+ 97	198+ 100
		null sp|DDF	79+ 29	80+ 28	239+ 89	238+ 90	241+ 90
		GCD	73+ 21	142+ 48	28+ 1	27+ 1	27+ 2
		total	450+ 167	522+ 193	463+ 189	460+ 188	466+ 192
17	8	linear facs	119+ 37	120+ 36			
		Q matrix	169+ 87	169+ 88	211+ 104	209+ 103	212+ 105
		null sp|DDF	86+ 34	87+ 35	296+ 115	293+ 115	295+ 115
		GCD	0+ 0	0+ 0	27+ 0	26+ 0	26+ 1
		total	397+ 159	400+ 159	534+ 219	528+ 218	533+ 221
19	1, 2, 2, 3	linear facs	153+ 49	154+ 50			
		Q matrix	171+ 82	174+ 84	213+ 103		180+ 62
		null sp|DDF	75+ 27	76+ 27	182+ 62		180+ 62
		GCD	357+ 125	198+ 72	315+ 74		636+ 286
		total	756+ 283	602+ 233	710+ 239		1027+ 451
41	1, 3, 4	linear facs	199+ 73	200+ 73			
		Q matrix	205+ 109	207+ 110	250+ 130	248+ 130	251+ 130
		null sp|DDF	79+ 28	79+ 28	250+ 92	248+ 93	250+ 91
		GCD	1150+ 444	136+ 43	28+ 2	27+ 2	27+ 2
		total	1633+ 654	622+ 254	528+ 224	523+ 225	528+ 223
103	3, 5	linear facs	228+ 100	229+ 99			
		Q matrix	162+ 88	163+ 89	311+ 171	305+ 165	309+ 167
		null sp|DDF	91+ 38	92+ 37	256+ 99	256+ 98	265+ 98
		GCD	1987+ 780	156+ 51	28+ 1	27+ 1	27+ 1
		total	2468+1006	640+ 276	595+ 271	588+ 264	601+ 266
1979	2, 6	linear facs	381+ 193	378+ 192			
		Q matrix	163+ 89	164+ 89	448+ 259	446+ 259	450+ 258
		null sp|DDF	100+ 39	100+ 39	273+ 93	278+ 92	272+ 93
		GCD	1140+ 329	224+ 41	28+ 2	27+ 2	26+ 1
		total	1784+ 650	866+ 361	749+ 354	751+ 353	748+ 352

表 3　$f_2(x)$ の因数分解の計算時間 (単位はマイクロ秒)

p	因子の次数	step	Berlekamp	Zassenhaus	DDF+Zass	C/Z	BenOr
			SPU+ VPU	SPU+VPU	SPU+VPU	SPU+VPU	SPU+VPU
7	全て	linear facs	137+ 51	138+ 51			
	3 次	Q matrix	228+ 135	231+ 135	268+ 157		372+ 161
		null sp|DDF	313+ 194	311+ 193	237+ 96		237+ 96
		GCD	1505+ 814	1306+ 737	1272+ 689		2174+ 1317
		total	2183+ 1194	1986+ 1116	1777+ 942		2783+ 1574
13	全て	linear facs	168+ 66	169+ 66			
	3 次	Q matrix	263+ 157	264+ 158	310+ 183		315+ 185
		null sp|DDF	296+ 176	294+ 174	256+ 105		257+ 104
		GCD	3028+ 1710	1658+ 951	1439+ 780		4558+ 2815
		total	3755+ 2109	2385+ 1349	2005+ 1068		5130+ 3104
19	全て	linear facs	216+ 86	218+ 87			
	3 次	Q matrix	600+ 406	613+ 412	647+ 429		672+ 449
		null sp|DDF	321+ 198	321+ 197	351+ 146		353+ 146
		GCD	1249+ 580	777+ 405	1185+ 626		3395+ 2217
		total	2386+ 1270	1929+ 1101	2183+ 1201		4420+ 2812
41	全て	linear facs	226+ 92	229+ 93			
	2 次	Q matrix	606+ 412	638+ 439	693+ 466	681+ 451	701+ 468
		null sp|DDF	296+ 184	298+ 185	184+ 69	182+ 68	185+ 68
		GCD	2839+ 1239	1037+ 476	1456+ 707	2692+ 1756	4932+ 3144
		total	3967+ 1927	2202+ 1193	2333+ 1242	3555+ 2275	5818+ 3680
103	全て	linear facs	322+ 147	325+ 148			
	3 次	Q matrix	625+ 439	627+ 434	765+ 517		773+ 518
		null sp|DDF	346+ 215	345+ 213	361+ 146		362+ 147
		GCD	8397+ 3646	826+ 414	1326+ 697		4532+ 3010
		total	9690+ 4447	2123+ 1209	2452+ 1360		5667+ 3675
1979	全て	linear facs	573+ 317	572+ 322			
	2 次	Q matrix	629+ 444	632+ 441	1002+ 691	1006+ 693	1021+ 698
		null sp|DDF	307+ 188	306+ 187	210+ 75	209+ 75	210+ 75
		GCD	138753+55820	1331+ 541	2029+ 944	6279+ 4355	8432+ 5502
		total	140272+56769	2841+ 1491	3241+ 1710	7494+ 5123	9663+ 6275
227951	全て	linear facs	900+ 545	929+ 572			
	2 次	Q matrix	722+ 434	641+ 451	1320+ 923		1433+ 921
		null sp|DDF	319+ 186	319+ 189	233+ 76		235+ 75
		GCD (ミリ秒)	14491+ 5310	3.641+1.806	4.382+2.374		533.7+338.5
		total (ミリ秒)	14493+ 5311	5.540+3.018	5.935+3.373		535.4+339.5

■実験結果に関して

以上の計測結果に関して，簡単なまとめを行う．

- 本稿では数値データを省略するが，行列 Q の構成と消去および DDF のステップではベクトル処理の効果は著しく，特に行列 Q に関するステップでは，上記 $n = 100$ の場合に，ベクトル処理を行った場合と行わない場合とでは計算時間に 10 倍程度の違いの生ずることが観測されている．
- その結果，ステップ毎の計算量の比較からだけでは判然としないが，実際の計算時間では GCD 計算の繰り返しが支配的である．
- この点で，確率的アルゴリズムである Cantor&Zassenhaus では，計算時間が乱数の生成に強く依存する一方，この点で Ben-Or アルゴリズムは安定している．その結果，[15] の結果と異なり，Ben-Or のアルゴリズムは有用と思われる．
- それ以上に注目すべきは，Zassenhaus のアルゴリズムがどの場合にも安定して速いことである．特に，Berlekamp のアルゴリズムより確実に高速であることが観測されているが，この点は，線形従属関係を導く際と線形因子への分解におけるベクトル処理が功を奏しているのであろう．
- Q の消去の計算量と DDF の (最悪の場合の) 計算量は同等だが，n が大きい場合，DDF の途中で多項式の次数が落ちていくことにより計算量が減ることがある．
- このことから，特に n が大きい場合，DDF の後 Zassenhaus のアルゴリズムを用いるという方法が有望と思われる．この方法も実際にインプリメントし，その計測結果を表中におさめた．しかし，DDF のために必要となる計算時間とメモリ所要量は必ずしも小さくなく，一般的な結論は出せない．

さらに Zassenhaus のアルゴリズムの優位性を確認するために，より高次の random に生成した多項式の因数分解も試みた．図 1 に，各アルゴリズムによる実行時間 (ベクトル処理を行った場合の SPU 時間) をグラフで示す．ここで用いた大部分の例では，全ての既約因子の次数は異なっており (確率的なアルゴリズムでは DDF までで因数分解が求まる)，同じ次数の因子があった場合でも高々 3 個である．これらの場合も Zassenhaus のアルゴリズムが高速であり，しかも注目すべきは，ここで試みた程度の次数では，全体の計算時間が，ベクトル処理の結果，計算量の n^3 から n^2 程度にまで減少して見える点である．このことをより明確にするために，ベクトル長 $(= n)$ の変化しない行列 Q の構成と消去の各ステップの計算時間を，ベクトル処理を行わない場合と行った場合 (の SPU 時間．VPU 時間は SPU 時間にほぼ比例) の両方について，図 2 にグラフとして示す．図では，ベクトル処理を行わなかった場合にグラフの傾きが計算量と一致しているのに対し，ベクトル処理を行った場合，あたかも計算量における n の指数が 1 程度減ったように見える．しかし，完全にベクトル化されているこれらの処理では，VPU において確実に n^3 又は n^2 個の要素が処理されており，その計算時間が n^2 又は n に比例することは説明がつかない．この点に関しては，田中輝雄氏 (日立中研) より次の指摘があった: ベクトル長がベクトル演算としては小さく，しかもループ中の $\mathbf{Z}_p$ 演算が極めて単純なため，ベクトル長に対する性能の立ち上がりが現れていない，即ち，計算時間中におけるオーバー

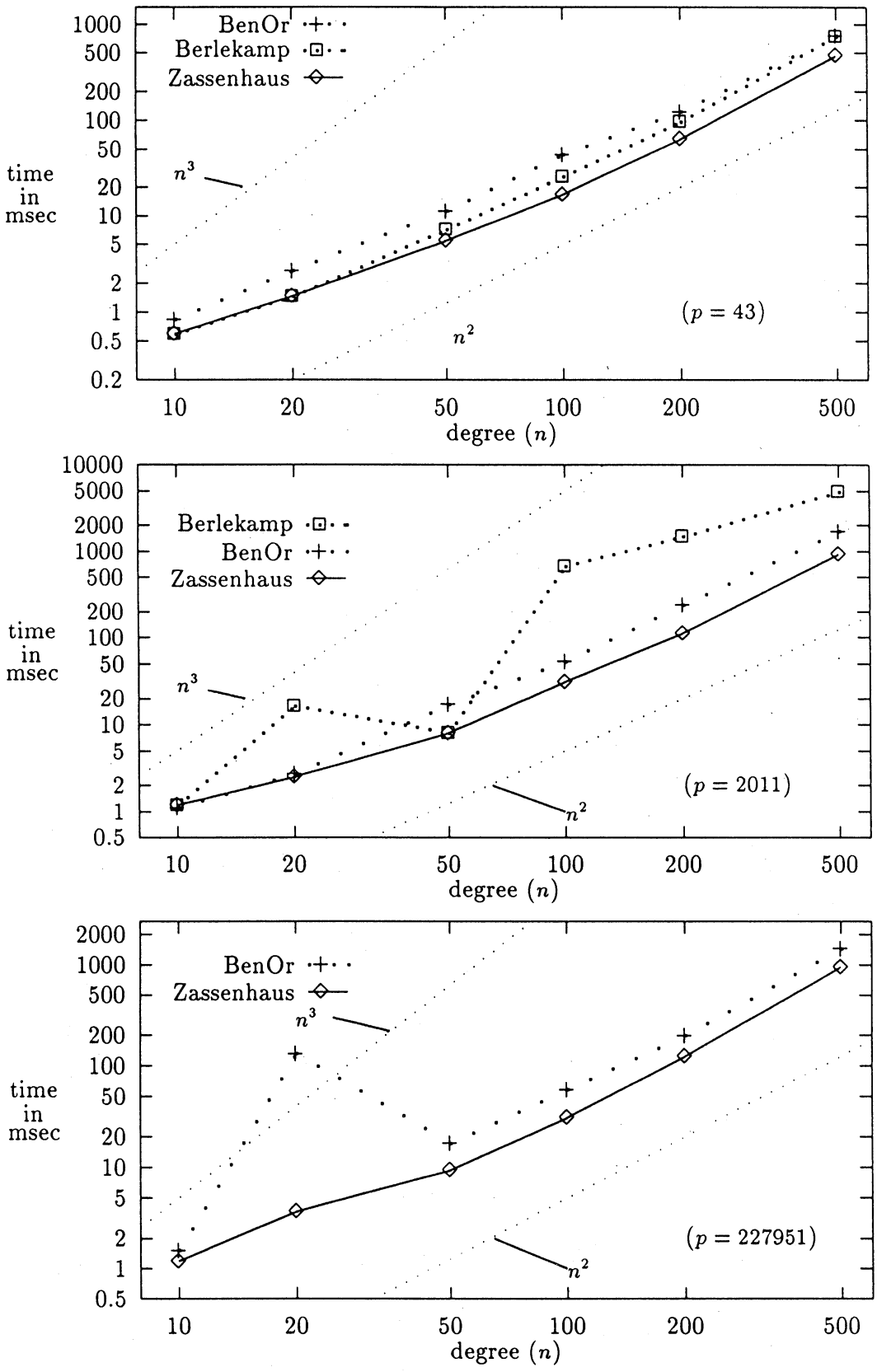
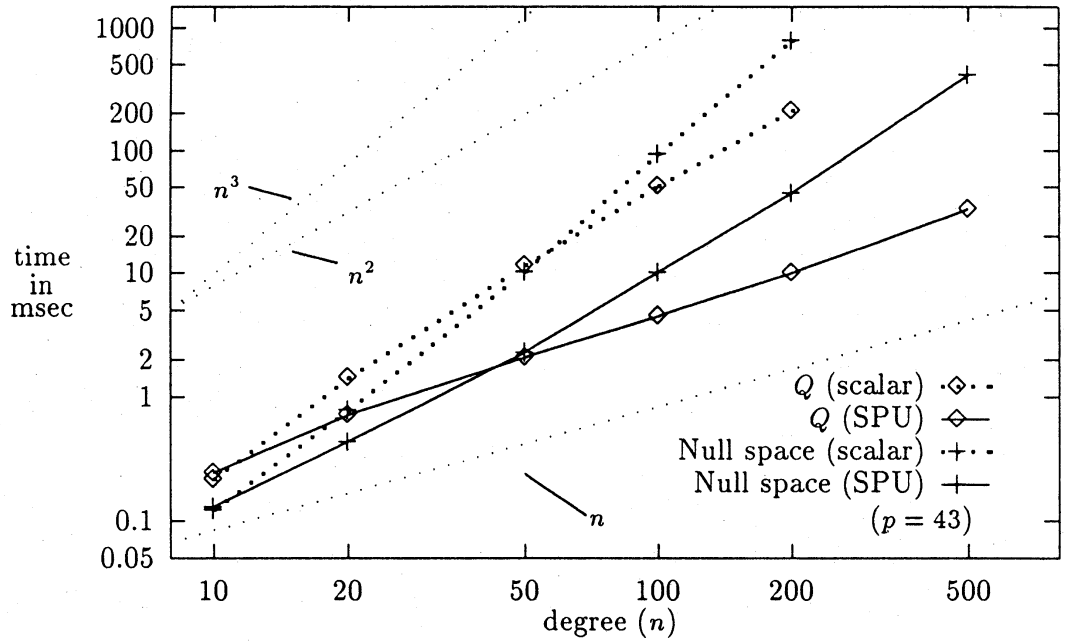


図 1　アルゴリズムによる実行時間の比較：時間 vs. 次数



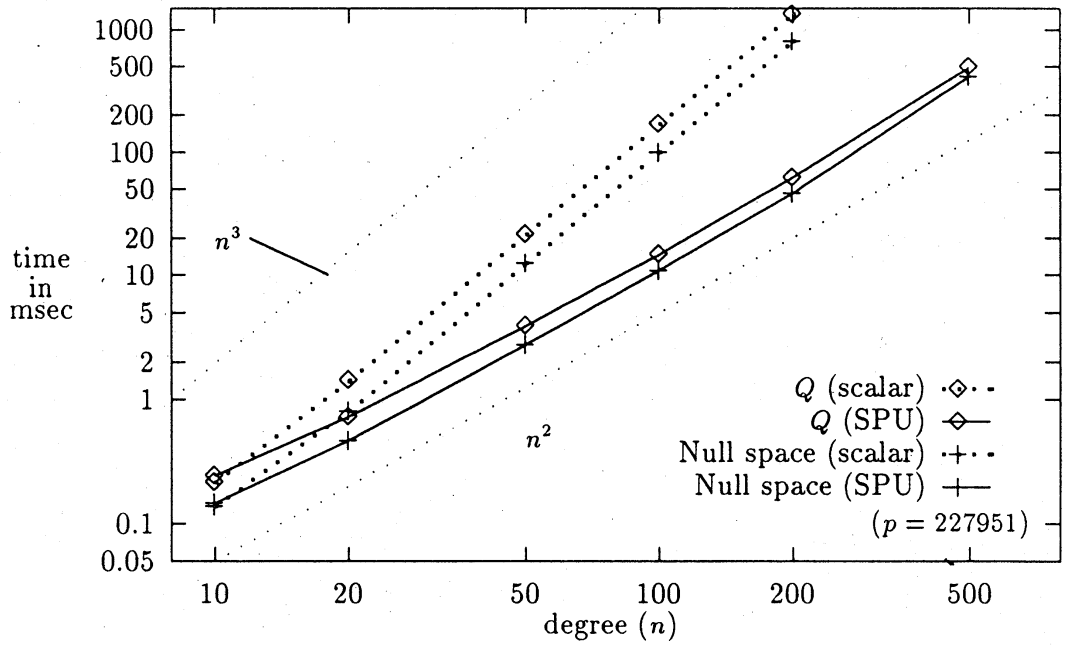


図 2　行列 Q の構成と消去の計算時間：時間 vs. 次数

ヘッドの割合が大きく，その傾きに演算回数が現れていないものと思われる．そこで，さらに高次の場合を試みた．問題としては，下記の [8] の例題を用いた (但し，$n = 1025, 2049, 4097$):

$$\left(x^{\lceil n/2 \rceil} + x + 1\right)\left(x^{\lfloor n/2 \rfloor} + x + 1\right) \mod p.$$

表 4 に，Zassenhaus のアルゴリズムによる計算時間を示す．表中には，計算量が正確に n^3 に比例

表 4　高次多項式の Zassenhaus のアルゴリズムによる計算時間 (単位はミリ秒)

n =次数 (n' =線形因子以外) $r_i = n'$の倍率 p	1025(1020) 1 32749	2049(2045) 2.0049 127	4097(4095) 4.0147 127
linear facs	31+ 23	36+ 31	122+ 113
Q matrix	3169+2973	325+ 308	602+ 584
null space ($\log_{r_i}$計算時間の比)	2697+2421 -	15013+13867 **2.468 2.509**	101461+96507 **2.609 2.651**
GCD	102+ 74	306+ 253	1176+ 1072
total	5998+5491	15680+14459	103361+98276

する零空間を求めるステップの計算時間の，次数への依存度 (次数の比による冪乗の指数) も示した．その数値を見る限り，この程度の大きな次数になると，$\log_{\text{次数}}$(計算時間) が 3 に近付いており，上記指摘の正しいことが確認される．いずれにせよ，特にベクトル処理を行った場合，従来から広く行われているような計算量及びそれによるアルゴリズムの比較が，あまり意味がないものと思われる．

15.6 並列処理の試み

高速化は，残すところ GCD 計算による因子の分離のみ (Zassenhaus では，$w(X) = 0$ の求根．冒頭に記したとおり，高速化の当初の目的に同じ) である．しかも，本稿に示したような問題では，計算時間は，次数よりも p の大きさに強く依存する．この分離は基本的に探索の問題であるから，並列処理による高速化が有望であろう．通常，前掲のアルゴリズムにおけるように，次式に基づく二分探索を行うのが普通だが，$p-1$ が因子 G を持つ場合，より一般的な式を用いることも可能である [10][4].

$$v(x)\left(v(x)^{(p-1)/2}-1\right)\left(v(x)^{(p-1)/2}+1\right) \equiv 0 \mod U(x) \text{ または } H_k(x),$$

ここでは，線形因子の積を分解する場合として，次式による Distinct Order Factorization について触れておく．G を $p-1$ の因子，ξ を 1 の原始 $(p-1)$ 乗根とする．

$$x^p - x \equiv \prod_{k=0}^{G-1}\left(x^{(p-1)/G} - \xi^{k(p-1)/G}\right) \mod p.$$

我々の行った実験は以下のとおり．詳細及び実験結果については，[12] や [18] を参照されたい．

- $k = 0, 1, \ldots, G-1$ のループを分割並列化し，
- 通信: あるプロセッサで non-trivial な $g = \gcd(w(x), x^{(p-1)/G} - \xi^{k(p-1)/G})$ が見つかった場合 g を通知し，受信した各プロセッサ上で，分解すべき多項式と $x^{(p-1)/G}$ とを g により簡約化．
- $p-1 = G_1 G_2 \cdots$ が多数の因子を持つ場合，上記の処理を，$G_1, G_2, \ldots$ 間でパイプライン化する．

15.7 今後の課題

ベクトル処理の試みについては，本稿で示した実測結果に見られるとおり，予想以上に高速なソフトウェア部品が完成し，また，特に次数 $\gg 1$ の場合に，ベクトル処理が非常に効果的であることが確認された．実際，従来は殆んど試みることすら諦められていたような規模の (高次の) 多項式の因数分解も，容易に得られるようになっている．この状況を踏まえ，今後さらに (1) 実用化，(2) さらに高次の多項式への対応，(3) 並列処理の進展等について，検討を続けていく予定である．

実用化にあたっては，$\mathbf{Z}$ 上での因数分解及び多変数の場合への拡張は必須であろう．これらについては，Hensel 構成のベクトル処理や並列処理を検討・実現していくことが必要であるし，また，極めて高次の多項式が扱えるようになったために，非常に多くの $\mathbf{Z}_p$ 上の因子が得られた場合の trial division の組合せの問題についても検討する必要が生じてきている．これについては，組合せの並列処理や，多項式計算量の L^3 アルゴリズムの適用も考慮する必要がある．また，多変数多項式に対しては，Kronecker のトリック等についても検討の必要があるかもしれない．さらに，こうして作られたソフトウェアを本当の意味で実用化するには，既存の数式処理システム (たとえば Risa/Asir) とも，プロセス間通信等の技術を用いて，連結・統合していく必要がある．より高次の多項式を扱う場合や並列処理については，アルゴリズムに関する研究が必要である．15.2 節に示した各種のアルゴリズムから明らかなように，因数分解は (1) 因子の個数の決定，(2) 適切な separating 多項式 ($\vec{v}$) の選択，(3) 探索 の 3 つからなる．この点に関し，これらの組合せとして未だ発表されていない様々なアルゴリズムが可能であることと，探索の部分において Zassenhaus のアルゴリズムが有用であることを指摘しておく．また，高次 ($n \gg 1$) の場合，並列処理のための算法として，反復法 (Wiedemann のアルゴリズム) に基づく [5] や，直接解法のインプリメント法についても検討しなければならない．これらは，主として (1) 及び (2) の段階を扱うものであるが，高次の場合には，何等かの方法により (必ずしも既約でない) 因子へと分解し，問題を break-down する必要がある．これに関しては，DDF を一般化した range decomposition [16] が有効と思われる．こうしたアルゴリズムの改良についても検討中である．

参考文献

[1]M. Ben-Or. Probabilistic algorithms in finite fields. In *Proc. 22nd IEEE Symposium on Foundation of Computer Science*, pp. 394–398, 1982.

[2]M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of 20th Symposium on the Theory of Computing*, pp. 301–309, Chicago, Illinois, May2–4 1988.

[3]E. R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967.

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

[5]E. Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Mathematics of Computation*, 1995. (to appear).

[6]E. Kaltofen and Y. Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Symbolic and Algebraic Computation, ISSAC '88*, number 358 in LNCS, pp. 467–474, Rome, Italy, July 4–8 1988. Springer-Verlag.

[7]E. Kaltofen, Y. N. Lakshman, and J.-M. Wiley. Modular rational sparse multivariate polynomial interpolation. In S. Watanabe and M. Nagata, editors, *Proceedings of ISSAC '90*, pp. 135–139, Tokyo, Japan, Aug. 20–24 1990.

[8]E. Kaltofen and A. Lobo. Factoring hig-degree polynomials by the black box Berlekamp algorithm. In J. von zur Gathen and M. Giesbrecht, editors, *Proceedings of ISSAC '94*, pp. 90–98, Oxford, England, July 20–22 1994.

[9]R. J. McEliece. Factorization of polynomials over finite fields. *Mathematics of Computation*, 23:861–867, 1969.

[10]R. T. Moenck. On the efficiency of algorithms for polynomial factoring. *Mathematics of Computation*, 31(137):235–250, Jan. 1977.

[11]H. Murao. Vectorization of symbolic determinant calculation. *SUPERCOMPUTER*, VIII(3):36–48, May 1991.

[12]H. Murao and T. Fujise. Modular algorithm for sparse multivariate polynomial interpolation and its parallel implementation. In H. Hong, editor, *Proceedings of PASCO '94*, pp. 304–315, Linz, Austria, Sept. 26–28 1994.

[13]A. C. Norman and J. Mitchel. Factorization in a functional language. In J. Della Dora and J. Fitch, editors, *Computer Algebra and Parallelism*, Computational Mathematics and Applications, pp. 133–141. Academic Press, 1989.

[14]V. Shoup. On the deterministic complexity of factoring polynomials over finite fields. *Information Processing Letters*, 33(5):261–267, Jan. 1990.

[15]V. Trevisan and P. Wang. Practical factorization of univariate polynomials over finite fields. In S. M. Watt, editor, *Proceedings of ISSAC '91*, pp. 22–31, Bonn, Germany, July 15–17 1991.

[16]J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.

[17]H. Zassenhaus. On Hensel factorization. *J. Number Theory*, 1:291–311, 1969.

[18]藤瀬. 並列論理型言語処理形 KLIC による PARI の並列化, 1985. (本講究録).