1997/11/4

数式処理 *J.JSSAC (2001)*
Vol. 8, No. 2, pp. 3 - 16

論文

On Checking Products of Modular Factors in Berlekamp-Hensel Type Factorization *

Yasuhiro Tsukada

Master's Program in Science and Engineering, University of Tsukuba

Tateaki Sasaki

Institute of Mathematics, University of Tsukuba

Tsukuba-shi, Ibaraki 305, Japan

Abstract

This article tests empirically two "dirty tricks" for the trial-division step of Berlekamp-Hensel type algorithm for the univariate polynomial factorization over **Z**. The tricks are 1) divisibility check of the constant term and 2) boundedness check of the second coefficient. So far, it has been said that 1) is quite effective but 2) is not so effective. However, defining the upper bound of the second coefficient suitably, we show by many examples that the trick 2) is also quite effective for polynomials of medium and large degrees, such as degree ≥ 15.

1 Introduction

The polynomial factorization, especially the univariate one over **Z**, is an old theme in computer algebra, but it still fascinates many researchers. Although an algorithm of polynomial-time complexity has been devised by Lenstra et al. [LLL82], the algorithm of Berlekamp-Hensel type [Ber67, Zas69] is very important practically. This algorithm consists of three steps: 1) factoring a given polynomial modulo a suitably chosen prime p, 2) Hensel-lifting of the modular factors, and 3) to find factors over **Z** by combining modular factors and performing the trial-division; for details, see **2**. Many researches have been made to speed up steps 1) and 2); for recent researches, see [M&F96] for step 1) and [C&E96] for step 2), for example.

*Work supported in part by Venture Business Lab., Univ. of Tsukuba.

From the viewpoint of computational time complexity, the last step (i.e., the trial-division step) is the most serious, and several effective methods have been proposed so far. The methods are simple and technical, hence they are called "dirty tricks". Among others, **1)** divisibility check of the constant term and **2)** boundedness check of the second coefficient are important. These checks are as follows. If $G(x) = g_n x^n + g_{n-1}x^{n-1} + \cdots + g_0$ is a factor over $\mathbf{Z}$, of $F(x) = f_m x^m + f_{m-1}x^{m-1} + \cdots + f_0$, then we have relations **1)** $g_0 | f_0$ and **2)** $|g_{n-1}| \leq B_1$, where B_1 is an upper-bound of the second coefficient of the factors of $F(x)$. Knuth [Knu69] mentions that these checks were suggested by G. E. Collins, and an old system MACSYMA [MAT77] employed the **check 1)** already in the 1970's.

So far, it has been said that the **check 1)** is quite effective, see for example [ABD85], but the **check 2)** is not so effective. In this article, by using a suitably defined upper-bound of B_1 and testing many examples, we show that the **check 2)** is also quite effective for polynomials of medium and large degrees. We also discuss why the **check 2)** is so effective and on rare cases in which the **check 2)** is ineffective.

2 Specification of factorization procedures

In this section, we specify Berlekamp-Hensel type algorithms for our empirical tests and give an upper-bound of B_1.

2.1 Skeleton of Berlekamp-Hensel type algorithm

Let $F(x), G(x) \in \mathbf{Z}[x]$, $F(x)$ be primitive and square-free, and $G(x)$ be a factor of $F(x)$, such that $\deg(G) \leq \deg(F)/2$:

$$F(x) = f_m x^m + f_{m-1}x^{m-1} + \cdots + f_0, \quad f_m > 0, \tag{1}$$

$$G(x) = g_n x^n + g_{n-1}x^{n-1} + \cdots + g_0, \quad n \leq m/2. \tag{2}$$

Below, by $\|F\|_2$ we denote the 2-norm of $F(x)$: $\|F\|_2 = (|f_m|^2 + |f_{m-1}|^2 + \cdots + |f_0|^2)^{1/2}$. The Berlekamp-Hensel type algorithm consists of three main steps, as follows.

Algorithm **Factorize-0**(F) ==

Comment : Skeleton of Berlekamp-Hensel type algorithm.

Input : $F(x)$, a primitive square-free polynomial over $\mathbf{Z}$.

Output : List of irreducible factors over $\mathbf{Z}$, of $F(x)$.

Step1 [Factorization over $\mathbf{Z}_p$] :

Select a small prime p which does not divide f_m and resultant$(F, \mathrm{d}F/\mathrm{d}x)$.

(Actually, put $p = 11$ first; if this prime is bad then put $p = 13$, and so on).

Factorize F/f_m over $\mathbf{Z}_p$ by Berlekamp's algorithm ($F_1^{(0)}, \ldots, F_r^{(0)}$ are monic).

$$F/f_m \equiv F_1^{(0)} \cdots F_r^{(0)} \pmod{p}. \tag{3}$$

Step2 [Hensel-lifting of the modular factors] :

Calculate Landau's bound B for the coefficients of G, see [Mig81,82].

$$|g_n| + |g_{n-1}| + \cdots + |g_0| \ \le\ 2^n \|F\|_2 \ \le\ 2^{m/2}\|F\|_2 \ \stackrel{\text{def}}{=}\ B. \tag{4}$$

Let k be the smallest integer satisfying $p^{k+1} \ge 2f_m B$, and perform the Hensel construction of F as follows ($F_1^{(k)}, \ldots, F_r^{(k)}$ are monic).

$$F \equiv f_m\, F_1^{(k)} \cdots F_r^{(k)} \pmod{p^{k+1}}. \tag{5}$$

Step3 [Combining modular factors and the trial-division] :

Put $S = \{F_1^{(k)}, \ldots, F_r^{(k)}\}$ and **F**actor**L**ist = nil.

For $s = 1, 2, \ldots, [r/2]$, do the following sub-steps **3.1** $\sim$ **3.4**.

3.1 Select a new combination of s elements $F_{i_1}^{(k)}, \ldots, F_{i_s}^{(k)}$ from S.

3.2 Calculate the product $\tilde{G} \equiv f_m\, F_{i_1}^{(k)} \cdots F_{i_s}^{(k)} \pmod{p^{k+1}}$.

3.3 If $\tilde{G}$ does not divide $(f_m F)$ over $\mathbf{Z}$ then go to **3.4**.
Save primitive-part($\tilde{G}$) into **F**actor**L**ist, and remove $F_{i_1}^{(k)}, \ldots, F_{i_s}^{(k)}$ from S.
If #elements(S) $< 2s$ then skip **3.4**.

3.4 If all the combinations of s elements are checked then $s \leftarrow s+1$.
If #elements(S) $\ge 2s$ then go to **3.1**.

If $S \neq \phi$ then calculate $\tilde{G} = f_m \times$ [product of the elements of S]
and save primitive-part($\tilde{G}$) into **F**actor**L**ist. □

Remark In the above procedure, the so-called distinct degree factorization step is omitted, which is practically quite effective. The reason is that we can see the effectiveness of **check**s **1)** and **2)** clearly by omitting such decorations.

2.2 Divisibility check of the constant term

Let $\tilde{G}(x)$ be a candidate of factor polynomial, computed in **Step3** :

$$\begin{aligned} \tilde{G} &\equiv f_m\, F_{i_1}^{(k)} \cdots F_{i_s}^{(k)} \pmod{p^{k+1}} &(6)\\ &\equiv f_m x^n + \tilde{g}_{n-1}x^{n-1} + \cdots + \tilde{g}_0 \pmod{p^{k+1}}. &(7) \end{aligned}$$

We can calculate the constant term $\tilde{g}_0$ in (7) quickly as

$$\begin{aligned} \tilde{g}_0 &\equiv f_m \prod_{j=1}^{s} [\text{ constant term of } F_{i_j}^{(k)}] \pmod{p^{k+1}} \\ &\Rightarrow \text{ normalize } \tilde{g}_0 \text{ into the interval } (-p^{k+1}/2,\ p^{k+1}/2). \end{aligned} \tag{8}$$

If $\tilde{g}_0$ does not divide $(f_m f_0)$ then $\tilde{G}(x)$ is not a factor of $f_m F(x)$ over $\mathbf{Z}$ and the value of $\tilde{g}_0$ will be distributed in the interval $(-p^{k+1}/2,\ p^{k+1}/2)$ widely. Therefore, this check will be quite effective. The reason is that the calculation of $\tilde{g}_0$ is much faster than the calculation of $\tilde{G}$ and the divisibility check of $(f_m f_0)$ by $\tilde{g}_0$ is quite cheap generally compared with the trial-division of $(f_m F)$ by $\tilde{G}$. (Note that, although the trial-division itself can be performed fast in many cases, we must construct every candidate $\tilde{G}$ in **Factorize-0**, which requires some amount of time.) We call the factorization procedure with this check **Factorize-1**.

Algorithm **Factorize-1**(F) ==

Comment : Perform the divisibility check of the constant term.

Comment : Replace the step **3.1** in **Factorize-0** by the following sub-steps.

3.1.1 Select a new combination of s elements $F_{i_1}^{(k)}, \ldots, F_{i_s}^{(k)}$ from S.

3.1.2 Calculate $\tilde{g}_0$ by formula (8).

3.1.3 If $\tilde{g}_0$ does not divide $(f_m f_0)$ then go to **3.4**. □

2.3 Boundedness check of the second coefficient

There is an upper-bound for the second coefficient of the factor $G(x)$, let it be B_1. Therefore, if $|\tilde{g}_{n-1}| > f_m B_1$, with $\tilde{g}_{n-1}$ given in (7), then $\tilde{G}(x)$ is not a factor of $f_m F(x)$. We can calculate $\tilde{g}_{n-1}$ quickly as

$$\begin{aligned} \tilde{g}_{n-1} &= \text{remainder}(\ f_m \sum_{j=1}^{s} [\text{second coefficient of } F_{i_j}^{(k)}],\ p^{k+1}\ ) \\ &\Rightarrow \text{ normalize } \tilde{g}_{n-1} \text{ into the interval } (-p^{k+1}/2,\ p^{k+1}/2). \end{aligned} \tag{9}$$

We call a factorization procedure with this check **Factorize-2**.

Algorithm **Factorize-2**(F) ==

Comment : Perform the boundedness check of the second coefficient.

Comment : Replace the step **3.1** in **Factorize-0** by the following sub-steps.

3.1.1 Select a new combination of s elements $F_{i_1}^{(k)}, \ldots, F_{i_s}^{(k)}$ from S.

3.1.2 Calculate $\tilde{g}_{n-1}$ by formula (9).

3.1.3 If $|\tilde{g}_{n-1}| > f_m B_1$ then go to **3.4**. □

2.4 Upper bounds of coefficients of a factor polynomial

The upper-bound B_1 being used so far is (see [Kal82], for example)

$$B_1 = m\|F\|_2 = (m/2^{m/2})B. \tag{10}$$

If the product $\tilde{G}$ computed in **Step3** is not a factor of $(f_m F)$, then $\tilde{g}_{n-1}$ in (7) will be distributed in the interval $(-p^{k+1}/2,\ p^{k+1}/2)$ widely. Therefore, if $B_1 \ll B$ then the **check 2)** will be quite effective. The bound B_1 in (10) is not extremely smaller than B for $m < 20$, hence we will define another bound B_1 below. (The following simple theorem will not be new, but we prove it for the convenience of readers).

Lemma 1

(see [1],[5]) *Let z be any zero-point of $F(x)$. Then, we have*

$$|z| \ \leq\ 1 + \frac{\max\{|f_{m-1}|, \ldots, |f_0|\}}{f_m} \ \stackrel{\text{def}}{=}\ \hat{z}_1, \tag{11}$$

$$|z| \ \leq\ \max\{\sqrt[j]{m|f_{m-j}|/f_m} \ \Big|\ j = 1, 2, \ldots, m\} \ \stackrel{\text{def}}{=}\ \hat{z}_2. \tag{12}$$

Theorem 2

The coefficient g_{n-i} of G in (2) is bounded as follows.

$$|g_{n-i}| \ \leq\ |g_n| \binom{n}{i} (\min\{\hat{z}_1, \hat{z}_2\})^i \ \leq\ f_m \binom{m/2}{i} (\min\{\hat{z}_1, \hat{z}_2\})^i \ \stackrel{\text{def}}{=}\ B_i \tag{13}$$

Proof Let the zero-points of $G(x)$ be $z_1, z_2, \ldots, z_n$, then we have

$$\begin{aligned} \frac{|g_{n-i}|}{|g_n|} &= |z_1 \cdots z_i + \cdots + z_{n-i+1} \cdots z_n| \\ &\leq |z_1 \cdots z_i| + \cdots + |z_{n-i+1} \cdots z_n|. \end{aligned}$$

Since $|z_j| \leq \hat{z} \ \stackrel{\text{def}}{=}\ \min\{\hat{z}_1, \hat{z}_2\}$ $(j = 1, 2, \ldots, n)$, and there are $\binom{n}{i}$ terms in the above right hand side, we have

$$\frac{|g_{n-i}|}{|g_n|} \leq \hat{z}^i + \cdots + \hat{z}^i = \binom{n}{i} \hat{z}^i.$$

Since $|g_n| \leq f_m$ and $n \leq m/2$, we finally obtain (13). □

Remark We had better define B_i as $B_i \ \stackrel{\text{def}}{=}\ f_m \binom{n}{i} (\min\{\hat{z}_1, \hat{z}_2\})^i$. However, since the value of n varies in the algorithm, we define B_i as in (13) so as to compare with B_1 in (10) simply.

Let us compare the bound B_1 in (10), let it be $B_{1,1}$, and the bound B_1 in (13), let it be $B_{1,2}$:

$$\begin{cases} B_{1,1} \ \stackrel{\text{def}}{=}\ m \times (f_m^2 + f_{m-1}^2 + \cdots + f_0^2)^{1/2}, \\ B_{1,2} \ \stackrel{\text{def}}{=}\ (m/2) f_m \times \min\{\hat{z}_1, \hat{z}_2\}. \end{cases} \tag{14}$$

Since both $B_{1,1}$ and $B_{1,2}$ vary largely as the coefficients of $F(x)$ change largely, we consider the following three models of polynomial $F(x)$, which are easy to analyze theoretically.

Below, we assume that $m \gg 1$, then the models show that the bound $B_{1,2}$ is a considerable improvement of the old bound $B_{1,1}$.

Model A : $f_m \simeq |f_{m-1}| \simeq \cdots \simeq |f_0|$.
We easily see that $\|F\|_2 \simeq ([m+1]f_m^2)^{1/2} \simeq (mf_m^2)^{1/2}$, $\hat{z}_1 \simeq 2$ and $\hat{z}_2 \simeq m$. Since $\hat{z}_1 < \hat{z}_2$ for $m \gg 1$, we have

$$\begin{cases} B_{1,1} & \simeq & m \times (mf_m^2)^{1/2} & = & m^{3/2} f_m, \\ B_{1,2} & \simeq & (m/2) f_m \times 2 & = & m f_m. \end{cases} \tag{15}$$

We see that $B_{1,2}/B_{1,1} \approx 1/\sqrt{m}$. For example, $B_{1,2}/B_{1,1}$ is about 0.22 for $m = 20$ and about 0.14 for $m = 50$.

Model B : $|f_{m-i}| \simeq \binom{m}{i} f_m$, $i = 1, 2, \ldots, m$.
Using identity $\sum_{i=0}^{m} \binom{m}{i}^2 = \binom{2m}{m}$, we find that $\|F\|_2 \simeq f_m \binom{2m}{m}^{1/2}$. Furthermore, we readily see that $\hat{z}_1 \simeq 1 + \binom{m}{m/2}$ and $\hat{z}_2 \simeq m^2$. Since $\hat{z}_1 > \hat{z}_2$ for $m > 9$, we use $\hat{z}_2$ for $B_{1,2}$, obtaining

$$\begin{cases} B_{1,1} & \simeq & m f_m \binom{2m}{m}^{1/2}, \\ B_{1,2} & \simeq & m^3 f_m / 2. \end{cases} \tag{16}$$

Using Stirling's formula, we find that $B_{1,2}/B_{1,1} \approx m^2 (\pi m)^{1/4} / 2^{m+1}$. For example, $B_{1,2}/B_{1,1}$ is about 0.00056 for $m = 20$ and about 4.0×10^{-12} for $m = 50$.

Model C : $|f_{m-i}| \simeq c^i f_m$, $i = 1, 2, \ldots, m$, with $c^{m-1} > m$ $(c > 1)$.
We see that $\hat{z}_1 \simeq 1 + c^m$ and $\hat{z}_2 \simeq \max\{mc, \sqrt[2]{mc^2}, \sqrt[3]{mc^3}, \cdots\} = mc$. Since $c^m > mc$ by assumption, we have $\hat{z}_2 < \hat{z}_1$, obtaining

$$\begin{cases} B_{1,1} & \simeq & m f_m \times (1 + c^2 + \cdots + c^{2m})^{1/2} & = & m f_m \left(\frac{c^{2m+2}-1}{c^2-1} \right)^{1/2}, \\ B_{1,2} & \simeq & (m/2) f_m \times mc & = & m^2 c f_m / 2. \end{cases} \tag{17}$$

Therefore, we find $B_{1,2}/B_{1,1} \approx m\sqrt{c^2-1}/(2c^m)$. For example, $B_{1,2}/B_{1,1} \approx 0.043$ for $c = 1.3$ and $m = 20$, $B_{1,2}/B_{1,1} \approx 0.098$ for $c = 1.1$ and $m = 50$, and $B_{1,2}/B_{1,1} \approx 0.0018$ for $c = 1.2$ and $m = 50$.

3 Empirical tests

Since the second coefficient $\tilde{g}_{n-1}$ will be distributed widely in the interval $(-p^{k+1}, p^{k+1}) \approx (-f_m B, f_m B)$, the **check 2)** is effective only if the following two conditions are satisfied.

$$\begin{cases} \text{Condition a)} & : & B_1 \ll B, \\ \text{Condition b)} & : & \text{Time[trial-division]} \gg \text{Time[\textbf{check 2)}}]. \end{cases} \tag{18}$$

Here, Time[trial-division] is the sum of times for $\tilde{G}$ computation and for the trial-division. On the other hand, (10) gives us $B_{1,1}/B = m/2^{m/2}$; for example

$$B_{1,1}/B \simeq 0.75, 0.5, 0.3125, 0.1093, 0.0195 \quad \text{for} \quad m = 6, 8, 10, 14, 20,$$

respectively. These values suggest that the **check 2)** with $B_{1,1}$ is not effective for $m \leq 10$. On the other hand, $B_{1,1}/B \simeq 0.0195, 3.8 \times 10^{-5}, 5.6 \times 10^{-8}, 7.3 \times 10^{-11}, 8.9 \times 10^{-14}$ for $m = 20, 40, 60, 80, 100$, respectively. Therefore, the **check 2)** with the old bound $B_{1,1}$ will be quite effective for polynomials of large degrees, such as $m \geq 20$. The improved bound $B_{1,2}$ will make the check more effective.

The above argument shows that Condition a) is critical for polynomials of degrees $10 \sim 20$. Therefore, we investigate the effectiveness of **check 2)** by applying **Factorize-0**, **-1**, and **-2** to polynomials of degrees $18 \sim 21$; we count the number of samples for which **check**s **1)** and **2)** fail, and measure Time[trial-division], Time[**check 1)**] and Time[**check 2)**]. Note that the effectiveness of **check 2)** does not depend much on the size of coefficients: the ratio $B_{1,1}/B$ is independent of f_m, and $B_{1,2}/B$ is also independent of f_m in the above three models. Therefore, we choose integers of $4 \sim 21$ figures as f_m.

Actually, the experiments were performed as follows. For each of **Model**s **A**, **B**, and **C**, we performed three tests. In each test, we generate 100 polynomials with coefficients distributed randomly, and apply **Factorize-0**, **-1** and **-2**. The polynomials used in each test are as follows.

Test A-1 Each sample is an irreducible polynomial of degree 20, with coefficients of random integers of 10 figures.

Test A-2 Each sample is the product of two irreducible polynomials of degree 10, with coefficients of random integers of 8 figures, hence the sample is of degree 20.

Test A-3 Each sample is the product of three irreducible polynomials of degrees 5, 8 and 8, with coefficients of random integers of 5, 8 and 8 figures, respectively, hence the sample is of degree 21.

Test B-1 Each sample is an irreducible polynomial of degree 20, where the coefficient of x^i term is a 10-figure random integer multiplied by ${}_{20}C_i$.

Test B-2 Each sample is the product of two irreducible polynomials of degrees 8 and 10, where the coefficients of x^i terms are random integers of 5 and 8 figures, respectively, multiplied by ${}_8C_i$ and ${}_{10}C_i$, respectively, hence the sample is of degree 18.

Test B-3 Each sample is the product of three irreducible polynomials of degrees 5, 5 and 8, where the coefficients x^i terms are random integers of 5, 8 and 8 figures, respectively, multiplied by ${}_5C_i$, ${}_5C_i$ and ${}_8C_i$, respectively, hence the sample is of degree 18.

Test C-1 Each sample is an irreducible polynomial of degree 20, with coefficients satisfying **Model C**: f_{20-i} is a random integer of about $[4+i/2]$ figures.

Test C-2 Each sample is the product of two irreducible polynomials of degree 10, with coefficients satisfying **Model C**: for each factor G, its coefficient g_{10-i} is a random integer of about $[5+i/2]$ figures.

Test C-3 Each sample is the product of three irreducible polynomials of degrees 5, 8 and 8, with coefficients satisfying **Model C**: for each factor G of degree n, its coefficient g_{n-i} is a random integer of about $[i+2]$ figures.

Tests **A-**i $(i=1,2,3)$ are for **Model A**, **Test**s **B-**i $(i=1,2,3)$ for **Model B**, and **Test**s **C-**i $(i=1,2,3)$ for **Model C**. The value of f_m is large in **Test A-**i and **Test B-**i, while it is small in **Test C-**i.

Results of these tests are shown in Tables A-i, B-i, and C-i $(i=1,2,3)$. In each table, we list the computation times spent in **Step3** of **Factorize-0**, **Factorize-1** and **Factorize-2**, where each timing datum for r $\mathbf{Z}_p$-factors is the average over all the samples having r modular factors. It should be emphasized that, *for all the samples we have tested, all the bad combinations of modular factors are detected by each of* **check**s **1)** *and* **2)**.

4 Explanation of the experimental data

Let us explain the experimental data in the previous section.

For Tables A-1, B-1, C-1. Each sample is an irreducuble polynomial of degree 20. Note that, if $F(x)$ is factorized into r irreducible factors over $\mathbf{Z}_p$, then we must check $2^{r-1}-1$ combinations of modular factors in **Step3**.

Since the **check**s **1)** and **2)** have detected all the bad combinations in our tests, there is no trial-division in **Factorize-1** and **-2**. Therefore, the timing data for **Factorize-1** and **-2** in Tables A-1, B-1 and C-1 are for the execution of sub-steps **3.1.1** $\sim$ **3.1.3**. We see that the timing data $T_{\rm check}(r)$, $4 \le r \le 7$, for **Factorize-1** and **-2** in Tables A-1, B-1 and C-1 are summarized very roughly as

$$T_{\rm check}(r) \approx C_{\rm C} \times (2^{r-1}-1) \text{ ms}, \quad \text{with } C_{\rm C} = 0.1. \tag{19}$$

On the other hand, the timing data $T_{\rm TryDiv}(r)$, $4 \le r \le 7$, for **Factorize-0** in Tables A-1, B-1 and C-1 are summarized very roughly as

$$T_{\rm TryDiv}(r) \approx C_{\rm T} \times [\#\text{ of } \tilde{G}\text{'s constructed}] \text{ ms}, \quad \text{with } C_{\rm T} = 2.0. \tag{20}$$

Here, the number of $\tilde{G}$'s constructed is $2^{r-1}-(r+1)$ in **Test**s **A-1**, **B-1** and **C-1**. This means that the time for trial-division is negligible compared with the time for $\tilde{G}$ computation.

Experiment 1: Each sample is an irreducible polynomial of degree 20.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
1	6	0.0	0.0	0.0
2	10	2.4	0.3	0.2
3	30	2.5	0.2	0.3
4	28	11.7	0.8	0.8
5	17	19.9	1.9	1.4
6	7	50.1	3.3	2.1
7	1	108.0	10.0	6.0
8	1	240.0	16.0	8.0

Table A-1: Results of **Test A-1**.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
1	4	0.0	0.0	0.0
2	12	2.5	0.3	0.3
3	39	2.6	0.4	0.4
4	21	12.0	0.7	0.7
5	13	20.0	1.9	1.3
6	8	52.9	3.1	2.0
7	2	112.0	10.0	6.0
8	1	189.0	16.0	8.0

Table B-1: Results of **Test B-1**.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
1	4	0.0	0.0	0.0
2	19	2.2	0.2	0.1
3	29	2.4	0.5	0.2
4	22	10.3	0.6	0.7
5	13	18.2	1.6	1.4
6	11	47.1	3.0	2.0
7	1	100.0	9.0	6.0
8	1	189.0	14.0	8.0

Table C-1: Results of **Test C-1**.

Experiment 2: Each sample is the product of 2 irreducible polynomials

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
2	2	4.5	5.0	5.0
3	8	6.0	4.9	5.0
4	19	13.2	5.9	5.8
5	27	18.2	6.8	6.5
6	27	41.0	8.9	8.0
7	12	66.9	11.6	9.6
8	4	184.8	20.5	13.8
9	1	110.0	20.0	15.0

Table A-2: Results of **Test A-2**.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
2	1	4.0	3.0	3.0
3	10	5.1	3.9	3.9
4	20	6.7	4.2	4.4
5	24	15.8	5.5	5.4
6	20	30.7	7.3	6.6
7	18	75.1	11.9	9.5
8	5	101.2	15.2	11.4
9	2	392.0	46.0	27.5

Table B-2: Results of **Test B-2**.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
2	3	4.0	3.0	3.7
3	5	4.8	4.0	3.6
4	14	8.8	4.4	4.3
5	31	14.5	5.3	5.2
6	23	36.3	7.3	6.3
7	18	66.0	10.4	8.5
8	5	152.2	16.4	11.0
9	1	407.0	43.0	24.0

Table C-2: Results of **Test C-2**.

Experiment 3: Each sample is the product of 3 irreducible polynomials

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
3	1	7.0	8.0	8.0
4	2	9.5	8.5	8.0
5	12	16.4	10.3	9.8
6	22	23.5	11.0	10.7
7	18	43.7	13.8	12.4
8	26	70.5	16.9	14.5
9	12	121.7	23.1	17.6
10	4	166.5	30.0	21.3
11	3	542.0	74.0	42.7

Table A-3: Results of **Test A-3**.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
3	1	6.0	7.0	7.0
4	4	8.0	7.5	7.0
5	16	12.1	8.2	7.8
6	29	19.4	9.4	9.1
7	29	33.2	11.3	10.2
8	10	55.7	15.2	12.5
9	10	141.3	26.7	18.3
11	1	668.0	119.0	56.0

Table B-3: Results of **Test B-3**.

number of $\mathbf{Z}_p$-factors	number of samples	**Factorize-0 Step3** (ms)	**Factorize-1 Step3** (ms)	**Factorize-2 Step3** (ms)
4	4	5.5	5.0	5.0
5	18	10.2	5.8	5.7
6	14	14.7	6.4	6.1
7	25	27.0	8.4	7.6
8	14	48.4	11.5	9.8
9	17	77.6	15.2	11.6
10	7	206.1	29.7	18.6
11	1	426.0	52.0	30.0

Table C-3: Results of **Test C-3**.

The reason of this fast trial-division is as follows. Consider the trial-division of $F'(x) = f'_l x^l + f'_{l-1}x^{l-1} + \cdots + f'_0$ by $\tilde{G}(x) = \tilde{g}_n x^n + \cdots + \tilde{g}_0$. If $\tilde{g}_n$ does not divide f'_l then the trial-division fails, otherwise we calculate $F'(x) - (f'_l/\tilde{g}_n)x^{l-n}\tilde{G}(x)$ and continue the leading term elimination further. If $\tilde{G}(x)$ does not divide $F'(x)$, we can usually detect it in early step of the leading term elimination.

It is interesting to note that both $T_{\rm check}$ and $T_{\rm TryDiv}$ are nearly the same in **Test**s **A-1**, **B-1** and **C-1**. This means that the timing data do not depend much on the polynomial models we have used but depend mostly on $\deg(F)$, f_m and r. In fact, although f_m is small in **Test C-1**, the sizes of the coefficients of $\tilde{G}(x)$ are not much different in these tests: the value of modulus p^{k+1} in (5) is about $10^{25}, 10^{28}$ and 10^{20} in **Test**s **A-1**, **B-1** and **C-1**, respectively.

Fitting (20) to the data in Tables A-1, B-1, and C-1 rather precisely, we see that the actual data are larger (resp. smaller) than (20) for smaller (resp. larger) values of r. The reason is that, if r is small (resp. large) then $\deg(\tilde{G})$ is averagely large (resp. small), hence the time for $\tilde{G}$ computation becomes large (resp. small).

For Tables A-2, B-2, C-2. Each sample is the product of two irreducible polynomials. Therefore, **Factorize-1** and **-2** perform one trial-division for each sample, hence one $\tilde{G}$ computation requires about 5ms, 4ms and 3.5ms in **Test**s **A-2**, **B-2** and **C-2**, respectively. Subtracting 5, 4 and 3.5 from the data for **Factorize-1** and **-2** in Tables A-2, B-2, and C-2, respectively, we see that the timing data $T_{\rm check}(r)$, $4 \le r \le 8$, for **Factorize-1** and **Factorize-2** can be fitted very roughly by (19). Similarly, the timing data $T_{\rm TryDiv}(r)$, $5 \le r \le 8$, for **Factorize-0** can be fitted very roughly by (20), with $C_{\rm T} = 1.5 \sim 1.3$.

For Tables A-3, B-3, C-3. Each sample is the product of three irreducible polynomials. Therefore, **Factorize-1** and **-2** perform two trial-divisions for each sample, hence one $\tilde{G}$ computation requires about 4ms, 3.5ms and 2.5ms in **Test**s **A-3**, **B-3** and **C-3**, respectively. Subtracting 2×4, 2×3.5 and 2×2.5 from the data for **Factorize-1** and **-2** in Tables A-3, B-3 and C-3, respectively, we see that the timing data $T_{\rm check}(r)$, $5 \le r \le 8$, for **Factorize-1** and **Factorize-2** can be fitted very roughly by (19), with $C_{\rm T} = 0.1 \sim 0.05$. However, the timing data $T_{\rm TryDiv}(r)$ for **Factorize-0** cannot be fitted well by (20). The reason is that, once an irreducible factor over **Z** have been detected, the number of combinations to be checked decreases largely, hence the number of $\tilde{G}$'s to be constructed changes largely from a sample to another.

5 Discussions

An optimal coding of the algorithm will reduce the timing data in Tables A-i, B-i and C-i considerably. However, the fact "*each of* **check**s **1)** *and* **2)** *detects all the bad*

combinations for all the samples we have tested" shows clearly that not only the **check 1)** but also the **check 2)** is quite effective practically for polynomials of degrees ≥ 15. Why has the **check 2)** been said not so effective so far. Probably, the reason is that people in the 1970's tested only polynomials of low degrees with the old bound $B_{1,1}$ for lack of the computer power.

It should be noted that the **check 2)** is not always effective: there are polynomials for which the **check 2)** is completely ineffective. For example, we can construct polynomials such that a) the second coefficients of all the modular factors are zero, b) all the modular factors are also the factors over $\mathbf{Z}$, and so on. However, we may say that such polynomials are in pathological cases, and the **check 2)** is quite effective in most cases. In such pathological cases, we must execute other checks such as the **check 1)** or the boundedness check of the third leading coefficient of $\tilde{G}$, etc.

Finally, let us consider what happens if $m = \deg(F)$ and q, the number of factors over $\mathbf{Z}$, become large. Note that, even if the **check**s **1)** and **2)** detect all the bad combinations, we must perform $q-1$ trial-divisions and q $\tilde{G}$ computations. Therefore, the computation times for **Step 3** in **Factorize-1** and **-2** will increase in proportion to q. The timing-data in the above tables show this fact clearly. However, the value of r will also increase as q increases, which requires us to check $O(2^{r-1})$ combinations of modular factors. Therefore, so long as $r \gg q$, the **check**s **1)** and **2)** are still quite effective. For larger $\deg(F)$, we may well expect more modular factors. Therefore, the **check**s **1)** and **2)** will usually become much more effective for polynomials of larger degrees.

Acknowledgments

The authors thank Dr. Motoyoshi of Electro-Technical Lab., Dr. Noro of Fujitsu Lab. and Prof. Abbott of Univ. Genova (Italy) for valuable information on "dirty tricks".

Note added in the revised manuscript

During incredibly long refereeing procedure (in fact, the authors got the referees' reports two years and half after the submission), we found a paper: *Factorization in* $\mathbf{Z}[x]$ *: The Searching Phase*, by J. Abbott, V. Shoup and P. Zimmermann (Proc. ISSAC'2000). The essence of this paper is to perform the **check 2)** very quickly by utilizing the floating-point arithmetic.

References

[ABD85] Abbott, J., Bradford, R.J. and Davenport, J.H.: A Remark on Factorization. SIGSAM Bulletin **19** (1985), pp. 31-33.

[Ber67] Berlekamp, E.R.: Factorizing polynomials over finite fields. Bell System Tech. J. **46** (1967), pp. 1853-1859.

[C&E96] Collins, G.E. and Encarnacion, M.J.: Improved techniques for factoring univariate polynomials. J. Symb. Comput. **21** (1996), pp. 313-327.

[Iri81] Iri, M.: *Numerical Computation* (in Japanese). Asakura Publishing Co., Tokyo, 1981, ch. 3.

[Kal82] Kaltofen, E.: Factorization of polynomial. *Computer Algebra : Symbolic and Algebraic Computation*, Springer-Verlag, Wien - New York, 1982, p. 101.

[Kun69] Knuth, D.E.: *The Art of Computer Programming*, Vol. 2 (Seminumerical Algorithms). Addison Wesley, 1969, Ch. 4.6.

[LLL82] Lenstra, Jr., A.K., Lenstra, H.W. and Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261** (1982), pp. 515-534.

[MAT77] MATHLAB group: *MACSYMA Reference Manual*. M.I.T.: Lab. Comput. Sci., 1977.

[M&F96] Murao, H. and Fujise, T.: Application of parallel processing to polynomial computation (in Japanese). J. JSSAC **5** (1996), pp. 2-17.

[Mig81] Mignotte, M.: Some inequalities about univariate polynomials. *Proc. of 1981 Symp. on Symbolic and Algebraic Computation*, ACM, 1981, pp. 195-199.

[Mig82] Mignotte, M.: Some useful bounds. *Computer Algebra : Symbolic and Algebraic Computation*, Springer-Verlag, Wien - New York, 1982, pp. 259-263.

[Tak81] Takagi, T.: *Lecture on Algebra* (in Japanese, revised ed.). Kyoritsu Publishing Co., Tokyo, 1981, ch. 3.

[Zas69] Zassenhaus, H.: On Hensel factorization – Part I. J. Number Theory **1** (1969), pp. 291-311.