A New Algorithm for Computing the Extended Hensel Construction of Multivariate Polynomials*

LU Dong · SUN Yao · WANG Dingkang

DOI:
Received: x x 20xx / Revised: x x 20xx

Abstract We present a new algorithm for computing the Extended Hensel Construction (EHC) of multivariate polynomials in main variable x and sub-variables $u_1, \ldots, u_m$ over a number field $\mathbb{K}$. This algorithm first constructs a set by using the resultant of two initial coprime factors w.r.t. x, and then obtains the Hensel factors by comparing the coefficients of x^i on both sides of an equation. Since the Hensel factors are polynomials of the main variable with coefficients in fraction field $\mathbb{K}(u_1, \ldots, u_m)$, the computation cost of handling rational functions can be high. Therefore, we use a method which multiplies resultant and removes the denominators of the rational functions. Unlike previously-developed algorithms that use interpolation functions (Moses and Yun [6]) or Gröbner basis (Sasaki and Inaba [9, 10]), Our algorithm rely little on polynomial division, and avoids multiplying by different factors when removing the denominators of Hensel factors. All algorithms are implemented using *Magma*, a computational algebra system and experiments indicate that our algorithm is more efficient than theirs.

Keywords Extended Hensel Construction, Multivariate polynomial, Sylvester matrix, Resultant.

1 Introduction

Being an important technique to deal with multivariate polynomials, Hensel construction has been applied to many branches of mathematics, such as polynomial factorization [3, 13], algebra equation solving [11], greatest common divisor computation [8] and other related areas. As a result, this task has drawn the interest of many researchers over the past several decades.

Let $\mathbb{K}$ be a number field and $\overline{\mathbb{K}}$ denote an algebraic closure of $\mathbb{K}$. Let $F(x, \mathbf{u})$ be a square-free polynomial in $\mathbb{K}[x, \mathbf{u}]$, $f(\mathbf{u})$ be the leading coefficient of $F(x, \mathbf{u})$ w.r.t. x, where $\mathbf{u}$ denotes the

LU Dong[1,2] · WANG Dingkang[1,2]
[1]KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China
Email: donglu@amss.ac.cn; dwang@mmrc.iss.ac.cn
SUN Yao
SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
Email: sunyao@iie.ac.cn

*This research was supported in part by the National Natural Science Foundation of China under Grant No. 11371356.

◇ *This paper was recommended for publication by Editor .*

sub-variables $u_1, \ldots, u_m$ ($m \geq 2$). We call $\vec{\alpha} \in \overline{\mathbb{K}}^m$ a singular point, if $F(x, \vec{\alpha})$ is not square-free. If $f(\vec{\alpha}) = 0$, then we say that the leading coefficient of $F(x, \mathbf{u})$ is singular at $\vec{\alpha}$. Musser first invented the Generalized Hensel Construction (GHC) to factorize multivariate polynomials in [7]. However, GHC fails for multivariate polynomials at a singular point or with singular leading coefficient. In both cases, we should shift $\vec{\alpha}$. Nevertheless, shifting $\vec{\alpha}$ often increases the number of terms and make the computation very time-consuming. This problem is known as the nonzero substitution problem in [2].

Since shifting $\vec{\alpha}$ leads to a dramatic increase of the computational costs, various attempts have been made to provide many different solutions to the nonzero substitution problem. Zippel in [14] attempted to solve the nonzero substitution problem by using Schwartz-Zippel's lemma. Unfortunately, this method requires the assumption that $F(x, \mathbf{u})$ is a monic polynomial, which is a major limitation. In [4], the authors present two new algorithms that extend Zippel's mehod to the case where $F(x, \mathbf{u})$ is not monic in the main variable x. A major breakthrough in Hensel construction was done by Sasaki and Kako [11], using a new method called Extended Hensel Construction (EHC). EHC can deal with the cases of multivariate polynomials at the singular point or with singular leading coefficient, without shifting $\vec{\alpha}$.

A major issue of the application of EHC is the computational efficiency of the Hensel factors of multivariate polynomials. This problem has been encountered by [6] and [9], who produced two useful algorithms. In [6], Moses and Yun proposed an algorithm which computes the Hensel factors of multivariate polynomials using interpolation functions. Moreover, Sasaki and Inaba in [9] enhanced the EHC of multivariate polynomials by using Gröbner bases. In this article, we focus on finding a new simple algorithm to compute Hensel factors efficiently. In the rest of the introduction, we informally sketch the main idea underlying this new algorithm, compare it with previous algorithms, and briefly describe its performance.

Main idea: It is well-known that the degree of Hensel factors w.r.t. x is bounded by the initial factors for any given multivariate polynomial, so we present our idea according to the proof of Proposition 9 in [1]. This proposition says: *Given two coprime polynomials $f, g \in \mathbb{K}[x, \mathbf{u}]$ of positive degree w.r.t. x, there are polynomials $A, B \in \mathbb{K}[x, \mathbf{u}]$ such that $\mathrm{res}_x(f, g) = Af + Bg$. Furthermore, $\deg_x(A) < \deg_x(g)$, $\deg_x(B) < \deg_x(f)$ and $\mathrm{res}_x(f, g) \in \mathbb{K}[\mathbf{u}]$, where $\mathrm{res}_x(f, g)$ denotes the* **resultant** *of f and g w.r.t. x.* We generalize the proposition to obtain a set $\{(A_i, B_i)\}_{i=0}^{N}$ by utilizing *Cramer's Rule*, where

$$x^i = \frac{1}{\mathrm{res}_x(f, g)} \cdot (A_i f + B_i g),$$

$A_i, B_i \in \mathbb{K}[x, \mathbf{u}]$ and $\deg_x(A_i) < \deg_x(g)$, $\deg_x(B_i) < \deg_x(f)$, $N = \deg_x(g) + \deg_x(f) - 1$. With the help of this set, Hensel factors can be efficiently calculated by comparing the coefficients of x^i on both sides of an algebraic equation. This step can save a lot of computation time. In order to avoid the high computational cost of handling rational functions, we remove the denominators of rational functions by introducing a new auxiliary variable $\mathscr{M}$. We treat $\mathscr{M}$ as $\frac{1}{\mathrm{res}_x(f,g)}$, and all computations are performed in $\mathbb{K}[\mathscr{M}, x, \mathbf{u}]$. If we combine the two steps mentioned above, we get a simple and efficient algorithm for computing the Hensel factors of multivariate polynomials.

Relation to the previous algorithms: The main idea of [6] is to construct some interpolation functions, and then convert them to the Hensel factors of multivariate polynomials. Moses and Yun used the extended Euclidean algorithm to compute the expression for x^i. They need to do many polynomial reductions in $\mathbb{K}(\mathbf{u})[x]$, and the computation is quite time-consuming. By contrast, we utilize the resultant of two initial coprime factors w.r.t. x to calculate the expressions for x^i. We only need to do some polynomial multiplication in $\mathbb{K}[x, \mathbf{u}]$, which can help us improve computational efficiency.

Sasaki and Inaba compute the Hensel factors of polynomials by using the Gröbner basis of two initial coprime factors in [9], and make various enhancements to improve the Gröbner basis computation in [10]. The computation of the syzygies of elements in Gröbner basis is a crucial step of their algorithm. This step is equivalent to the computation of the expressions of x^i in our algorithm. Moreover, their method requires the introduction of many different auxiliary variables to avoid dealing with fractional functions, when polynomials are reduced to nonzero by using Gröbner basis. However, our approach performs all calculations in a polynomial ring by introducing a auxiliary variable.

Performance: Without loss of generality, we assume that the arithmetic operations run in constant time. Our algorithm, as well as Moses and Yun's (MY), and Sasaki and Inaba's (SI) algorithms have all been implemented on *Magma* computational algebra system. Experiments indicate that our algorithm is faster than theirs. The codes (EHC Package) of our algorithm are available on the web: `http://www.mmrc.iss.ac.cn/~dwang/software.html`.

This paper is structured as follows: In Section 2, we introduce the EHC of multivariate polynomial, and present the problem that we shall consider. Section 3 contains our main theoretical results and a new efficient algorithm. Furthermore, an enhancement which improves our algorithm is described. In Section 4, we implement the three algorithms mentioned above and compare the performance of these algorithms by using three examples. Finally, Section 5 includes a conclusion as well as a discussion about the influence of the number of sub-variables between our algorithm and SI algorithm, and a future research direction.

2 Preliminaries

We treat x, $\mathbf{u}$ as the main variable and the sub-variables, respectively, where $\mathbf{u} = u_1, \ldots, u_m$ ($m \geq 2$). Let $\mathbb{K}[\mathbf{u}]$, $\mathbb{K}(\mathbf{u})$ and $\mathbb{K}\{(\mathbf{u})\}$ be the ring of polynomials, the field of rational functions and the ring of formal power series of rational functions, respectively, over $\mathbb{K}$, in sub-variables $\mathbf{u}$. Let F, G and H be three polynomials in $\mathbb{K}[x, \mathbf{u}]$. By $\deg_x(F)$ and $\mathrm{res}_x(G, H)$, we denote the degree of F and the resultant of G and H, respectively, w.r.t. x. Let $\mathrm{rem}(G, H)$ be the remainder of G and H.

Next, we first review some useful notions which play a central role in EHC, and then introduce the procedure of EHC. Finally, we present the problem we are considering.

2.1 Newton Line and Newton Polynomial

Definition 2.1 Let $F(x, \mathbf{u}) \in \mathbb{K}[x, \mathbf{u}]$. For each nonzero term $cx^d t^e u_1^{e_1} \cdots u_m^{e_m}$ of $F(x, t\mathbf{u})$, plot a dot at the point (d, e) in the descartes coordinate system (X, Y)-plane, where t is the total-degree variables for $\mathbf{u}$ and $e = e_1 + \cdots + e_m$. The Newton polygon $\mathcal{N}$ of $F(x, \mathbf{u})$ is a convex hull containing all the dots plotted. Let the lower sides of $\mathcal{N}$, traced clockwise, be $\mathcal{N}_1, \ldots, \mathcal{N}_p$ which we call Newton lines. For each $i \in \{1, \ldots, p\}$, the Newton polynomial $\overline{F}_{\mathcal{N}_i}(x, \mathbf{u})$ is the sum of all the terms of $F(x, \mathbf{u})$ plotted on $\mathcal{N}_i$.

For example, let $F = x^3(u_1^2+u_2^2+u_2^3)+x^2(2u_1^2-u_2^2+u_1^3)+x(u_1-u_2+2u_2^2)+3u_2^2+2u_1-1$. Then the Newton Line $\mathcal{N}$ of F is as follows, and the Newton polynomial $\overline{F}_{\mathcal{N}}$ of F is $x^3(u_1^2 + u_2^2) - 1$.

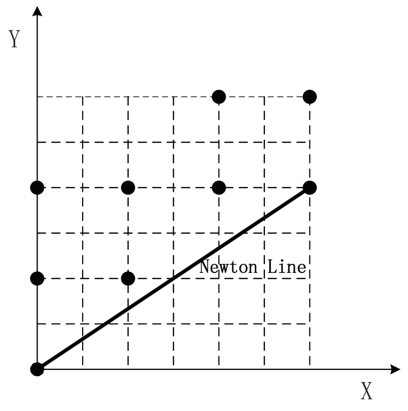

Figure 1 The Newton Line of F

Remark 2.2 When $F(x, \mathbf{u}) \in \mathbb{K}\{(\mathbf{u})\}[x]$, the definitions of Newton polygon, Newton line and Newton polynomial of $F(x, \mathbf{u})$ are similar to the Definition 2.1. We refer [3] for more detials.

2.2 The Procedure of EHC

For any given square-free polynomial $F(x, \mathbf{u})$, the EHC of $F(x, \mathbf{u})$ is performed successively on $\mathcal{N}_1 \to \mathcal{N}_2 \to \cdots \to \mathcal{N}_p$ (see [3]). Let (n_0, v_0) and (n_i, v_i) be the coordinate of right end of $\mathcal{N}_1$ and left end of $\mathcal{N}_i$, respectively, where $1 \leq i \leq p$. Then the slope of $\mathcal{N}_i$ is given by $\lambda_i = (v_{i-1} - v_i)/(n_{i-1} - n_i)$. Let $\widehat{n}_i$ and $\widehat{v}_i$ be integers satisfying $\widehat{n}_i > 0$, $\lambda_i = \widehat{v}_i/\widehat{n}_i$ and $\gcd(\widehat{n}_i, \widehat{v}_i) = 1$. Then we define $\mathcal{F}_i(x, \mathbf{u}, t)$ as follows:

$$\mathcal{F}_i(x, \mathbf{u}, t) \triangleq t^{\widehat{n}_i(\lambda_i n_i - v_i)} F(x/t^{\widehat{v}_i}, t^{\widehat{n}_i}\mathbf{u}). \tag{1}$$

Assume that the Newton polynomial of $F(x, \mathbf{u})$ on $\mathcal{N}_i$ is $\overline{F}_{\mathcal{N}_i}(x, \mathbf{u})$. We use a conventional method to factorize $\overline{F}_{\mathcal{N}_i}(x, \mathbf{u})$ in $\mathbb{K}[x, \mathbf{u}]$ as follows:

$$\begin{cases} \overline{F}_{\mathcal{N}_i}(x, \mathbf{u}) = G_{i1}^{(0)}(x, \mathbf{u}) \cdots G_{ir_i}^{(0)}(x, \mathbf{u}), \ \ r_i \geq 2 \\ \gcd(G_{ij}^{(0)}, G_{il}^{(0)}) = 1 \ \ for \ \ any \ \ j \neq l. \end{cases} \tag{2}$$

Note that, if $\overline{F}_{\mathcal{N}_i}(x, \mathbf{u})$ is irreducible, then $F(x, \mathbf{u})$ is also irreducible. According to some iterative relations, we can construct $G_{ij}^{(k)}(x, \mathbf{u}, t)$ $(j = 1, \ldots, r_i)$, $k = 0 \to 1 \to 2 \to \cdots$, such that

$$\mathcal{F}_i(x,\mathbf{u},t) \equiv G_{i1}^{(k)}(x,\mathbf{u},t)\cdots G_{ir_i}^{(k)}(x,\mathbf{u},t) \pmod{t^{k+1}}. \tag{3}$$

We refer to [11] for more details of the construction procedure of $G_{ij}^{(k)}(x,\mathbf{u},t)$. Let $t=1$, then $G_{ij}^{(k)}(x,\mathbf{u},1)$ is called a **Hensel factor** of $F(x,\mathbf{u})$ corresponding to $G_{ij}^{(0)}(x,\mathbf{u})$, where $G_{ij}^{(k)}(x,\mathbf{u},1) \in \mathbb{K}(\mathbf{u})[x]$, $k=1,2,\ldots$. The above steps are the procedure of EHC. By applying the EHC to $F(x,\mathbf{u})$ w.r.t. $\mathcal{N}_i$ infinite times with initial factors $G_{ij}^{(0)}(x,\mathbf{u})$ $(j=1,\ldots,r_i)$, $F(x,\mathbf{u})$ can be factorized as

$$F(x,\mathbf{u}) = G_{i1}^{(\infty)}(x,\mathbf{u})\cdots G_{ir_i}^{(\infty)}(x,\mathbf{u}), \tag{4}$$

where $G_{ij}^{(\infty)}(x,\mathbf{u}) \triangleq G_{ij}^{(\infty)}(x,\mathbf{u},1)$, and $G_{ij}^{(\infty)}(x,\mathbf{u}) \in \mathbb{K}\{(\mathbf{u})\}[x]$. If $G_{ij}^{(\infty)}(x,\mathbf{u})$ contain reducible factors in $\mathbb{K}[x,\mathbf{u}]$, then we can factorize $G_{ij}^{(\infty)}(x,\mathbf{u})$ further. We factorize the Newton polynomial for $G_{ij}^{(\infty)}(x,\mathbf{u})$ similarly, and continue the above procedure.

2.3 The Problem

In this paper, we focus on how to quickly compute the Hensel factors $\{G_{ij}^{(k)}(x,\mathbf{u},1)\}_{j=1}^{r_i}$ $(k=1,2,\ldots)$ of $F(x,\mathbf{u})$ on Newton line $\mathcal{N}_i$. Without loss of generality, we can assume that the square-free polynomial $F(x,\mathbf{u})$ has only one Newton line $\mathcal{N}$. Moreover, each Hensel factor of $F(x,\mathbf{u})$ is mutually prime, and computed independently from others. Therefore, it is enough to discuss the case of two initial coprime factors, i.e., $\overline{F}_{\mathcal{N}}(x,\mathbf{u}) = G_0(x,\mathbf{u})H_0(x,\mathbf{u})$, where $G_0, H_0 \in \mathbb{K}[x,\mathbf{u}]$. Then, the formulas on EHC become as follows.

$$\begin{cases} \mathcal{F}(x,\mathbf{u},t) \equiv \mathcal{G}^{(k)}\mathcal{H}^{(k)} \pmod{t^{k+1}}, \\ t^k \cdot \delta F^{(k)} \equiv \mathcal{F} - \mathcal{G}^{(k-1)}\mathcal{H}^{(k-1)} \pmod{t^{k+1}}. \end{cases} \tag{5}$$

These formulas must satisfy the following relations.

$$\begin{cases} \delta F^{(k)} = \delta H^{(k)} G_0 + \delta G^{(k)} H_0, \\ \mathcal{G}^{(k)} = \mathcal{G}^{(k-1)} + t^k \cdot \delta G^{(k)}, \\ \mathcal{H}^{(k)} = \mathcal{H}^{(k-1)} + t^k \cdot \delta H^{(k)}, \\ \mathcal{G}^{(0)} = G_0, \mathcal{H}^{(0)} = H_0, \\ \deg_x(\delta H^{(k)}) < \deg_x(H_0), \\ \deg_x(\delta G^{(k)}) < \deg_x(G_0). \end{cases} \tag{6}$$

According to the above formulas and relations, our goal is to quickly compute the Hensel factors of $F(x,\mathbf{u})$: $G^{(k)}$ and $H^{(k)}$, where $G^{(k)} \triangleq \mathcal{G}^{(k)}(x,\mathbf{u},1)$, $H^{(k)} \triangleq \mathcal{H}^{(k)}(x,\mathbf{u},1)$, $k=1,2,\ldots$. When $k=\infty$, we can factorize $F(x,\mathbf{u})$ into the following form:

$$F(x,\mathbf{u}) = G^{(\infty)}(x,\mathbf{u})H^{(\infty)}(x,\mathbf{u}), \tag{7}$$

where $G^{(\infty)}(x,\mathbf{u}), H^{(\infty)}(x,\mathbf{u}) \in \mathbb{K}\{(\mathbf{u})\}[x]$.

From the equation (6), it can be shown that $\deg_x(H^{(k)}) = \deg_x(H_0)$ and $\deg_x(G^{(k)}) = \deg_x(G_0)$ for every k. Note that $\delta H^{(k)}, \delta G^{(k)} \in \mathbb{K}(\mathbf{u})[x]$ in general, will take a vast amount of processing time to deal with the complex fractional functions. Furthermore, another difficulty in calculating the Hensel factors of $F(x, \mathbf{u})$ lies to the degree constraint w.r.t. x. Therefore, it is essential to present a new algorithm for computing the Hensel factors $G^{(k)}, H^{(k)}$ faster.

3 Main Results

In this section, we first propose a method to efficiently compute $\delta H^{(k)}$ and $\delta G^{(k)}$, then we deal with the problem of degree constraint w.r.t. x, and finally we remove the denominators of rational functions by multiplying the resultant of two initial factors w.r.t. x. We also propose an enhancement to improve our new algorithm.

3.1 EHC Based on Resultant

Assume that $\overline{F}_{\mathcal{N}}(x, \mathbf{u}) = G_0(x, \mathbf{u})H_0(x, \mathbf{u})$, where $G_0, H_0 \in \mathbb{K}[x, \mathbf{u}]$ have positive degree w.r.t. x. Let

$$\begin{cases} G_0 = a_l x^l + a_{l-1}x^{l-1} + \cdots + a_0, & a_l \neq 0, \\ H_0 = b_n x^n + b_{n-1}x^{n-1} + \cdots + a_0, & b_n \neq 0, \end{cases} \tag{8}$$

where $a_i, b_j \in \mathbb{K}[\mathbf{u}]$. Then the **Sylvester** matrix of G_0 and H_0 w.r.t. x, denoted $\mathrm{Syl}(G_0, H_0, x)$ is a coefficient matrix of the system of equations given in (8). Thus, $\mathrm{Syl}(G_0, H_0, x)$ is the following $(n + l) \times (n + l)$ matrix:

$$\begin{pmatrix}
a_l & a_{l-1} & \cdots & a_0 & & & & \\
 & a_l & a_{l-1} & \cdots & a_0 & & & \\
 & & \ddots & \ddots & & & \ddots & \\
 & & & & a_l & a_{l-1} & \cdots & a_0 \\
b_n & b_{n-1} & \cdots & \cdots & b_0 & & & \\
 & b_n & b_{n-1} & \cdots & \cdots & b_0 & & \\
 & & \ddots & \ddots & & & \ddots & \\
 & & & b_n & b_{n-1} & \cdots & \cdots & b_0
\end{pmatrix},$$

where the empty spaces are filled by zeros. The **resultant** of G_0 and H_0 w.r.t. x is the determinant of the Sylvester matrix. Thus,

$$\mathrm{res}_x(G_0, H_0) = \det(\mathrm{Syl}(G_0, H_0, x)).$$

We generalize the proposition 9 in [1] to obtain the following proposition.

Proposition 3.1 *Assume that G_0 and H_0 are defined as above, there are polynomials $A_i, B_i \in \mathbb{K}[x, \mathbf{u}]$ such that*

$$x^i \cdot \mathrm{res}_x(G_0, H_0) = A_i G_0 + B_i H_0, \tag{9}$$

where $\deg_x(A_i) < n$, $\deg_x(B_i) < l$ *and* $0 \le i \le n + l - 1$.

Proof Note that

$$\mathrm{Syl}(G_0, H_0, x) \cdot \begin{pmatrix} x^{l+n-1} \\ x^{l+n-2} \\ \vdots \\ x^{l} \\ x^{l-1} \\ x^{l-2} \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} x^{n-1}G_0 \\ x^{n-2}G_0 \\ \vdots \\ G_0 \\ x^{l-1}H_0 \\ x^{l-2}H_0 \\ \vdots \\ H_0 \end{pmatrix}. \tag{10}$$

Let $y_i = x^i$, then Equation (10) can be solved as a linear system of equations. Assume that $\vec{w} = [x^{n-1}G_0,\ x^{n-2}G_0, \cdots, G_0,\ x^{l-1}H_0, x^{l-2}H_0, \cdots, H_0]^T$ is an $(n+l)$-dimensional column vector, where superscript ^{T} denotes transposition. By $\mathrm{Syl}_i(G_0, H_0, x)$, we denote the matrix where the i-th column of the $\mathrm{Syl}(G_0, H_0, x)$ has been replaced by $\vec{w}$. According to *Cramer's Rule*, it follows that

$$y_i \cdot \mathrm{res}_x(G_0, H_0) = \det(\mathrm{Syl}_{n+l-i}(G_0, H_0, x)).$$

Expanding the $(n+l-i)$-th column of $\mathrm{Syl}_i(G_0, H_0, x)$, we have $A_i, B_i \in \mathbb{K}[x, \mathbf{u}]$ such that Equation (9) holds, and $\deg_x(A_i) < n$, $\deg_x(B_i) < l$. ∎

Remark 3.2 The result of Proposition 3.1 has been used frequently in the literature. For example, our idea is similar to that in [12]. Sasaki T and Sasaki M compute the expression of x^i by using the extended Euclidean algorithm, and do many polynomial reduction in $K(\mathbf{u})[x]$. Nevertheless, we utilize the resultant of two initial coprime factors w.r.t. x to calculate the expressions for x^i and do some polynomial multiplication in $K[x, \mathbf{u}]$.

Since $\deg_x(\delta H^{(k)}) < \deg_x(H_0)$ and $\deg_x(\delta G^{(k)}) < \deg_x(G_0)$, we obtain $0 \le \deg_x(\delta F^{(k)}) \le n+l-1$. We write $\delta F^{(k)}$ in the form

$$\delta F^{(k)} = c_{n+l-1}^{(k)} x^{n+l-1} + c_{n+l-2}^{(k)} x^{n+l-2} + \cdots + c_0^{(k)},$$

where $c_i^{(k)} \in \mathbb{K}(\mathbf{u})$. From Proposition 3.1 we have

$$\delta F^{(k)} = \frac{(\sum_{i=0}^{n+l-1} c_i^{(k)} A_i)G_0 + (\sum_{i=0}^{n+l-1} c_i^{(k)} B_i)H_0}{\mathrm{res}_x(G_0, H_0)}. \tag{11}$$

We introduce an auxiliary variable $\mathscr{M}$ to avoid dealing with fractional functions. Substitute $\mathscr{M}$ for $\frac{1}{\mathrm{res}_x(G_0,H_0)}$ in $\delta F^{(k)}$, and we get

$$\begin{cases} \delta H^{(k)} = \mathscr{M}(\sum_{i=0}^{n+l-1} c_i^{(k)} A_i), \\ \delta G^{(k)} = \mathscr{M}(\sum_{i=0}^{n+l-1} c_i^{(k)} B_i). \end{cases} \tag{12}$$

According to the equation (6), the Hensel factors $G^{(k)}$ and $H^{(k)}$ of $F(x, \mathbf{u})$ for every k are

$$\begin{cases} G^{(k)} = G^{(k-1)} + \mathscr{M}(\sum_{i=0}^{n+l-1} c_i^{(k)} A_i), \\ H^{(k)} = H^{(k-1)} + \mathscr{M}(\sum_{i=0}^{n+l-1} c_i^{(k)} B_i). \end{cases} \tag{13}$$

3.2 A New Algorithm

Proceeding as in the idea of Subsection 3.1, we have a new algorithm to compute the Hensel factors $G^{(k)}$ and $H^{(k)}$ of $F(x, \mathbf{u})$.

Input: A square-free polynomial $F(x, \mathbf{u})$.

Output: The Hensel factors $G^{(k)}$ and $H^{(k)}$ of $F(x, \mathbf{u})$.

Step 1: Compute the Newton polynomial $\overline{F}_{\mathcal{N}}$ of F.

Step 2: Factorize $\overline{F}_{\mathcal{N}}$ in $\mathbb{K}[x, \mathbf{u}]$: $\overline{F}_{\mathcal{N}} = G_0 H_0$, where G_0 and H_0 are prime.

Step 3: Calculate the set $\{(A_i, B_i)\}_{i=0}^{n+l-1}$.

Step 4: Introduce an auxiliary variable $\mathscr{M}$, and for $k = 1 \to 2 \to 3 \to \cdots$,

Step 4.1: Compute $\delta H^{(k)}$ and $\delta G^{(k)}$ in $\mathbb{K}[\mathscr{M}, x, \mathbf{u}]$ by using the Equation (12);

Step 4.2: Compute $G^{(k)}, H^{(k)}$ by using the Equation (13);

We use an example to illustrate the specific calculation process of our algorithm.

Example 3.3 Let $F(x, u_1, u_2) = u_1^2 u_2 x^4 + (u_1^3 u_2^2 + u_1 u_2^2 + u_1 u_2)x^3 + (u_1^4 u_2 + u_1^2 u_2^3 + 2u_1^2 u_2^2 + u_1 - u_2)x^2 + (u_1^3 u_2^2 + u_1^3 u_2 + u_1^2 u_2 + u_1 u_2^3 + 3u_1 u_2)x + u_1^3 - u_1^2 u_2 + u_1 u_2 - u_2^2$.

F has two Newton lines of slopes: $\lambda_1 = 1$ and $\lambda_2 = -\frac{1}{2}$. We only calculate the Hensel factors of F on $\mathcal{N}_1$. The Newton Polynomial $\overline{F}_{\mathcal{N}_1}$ is $(x^2 u_1^2 u_2 + x u_1 u_2 + u_1 - u_2)x^2$. Let $G_0 = x^2 u_1^2 u_2 + x u_1 u_2 + u_1 - u_2$ and $H_0 = x^2$. Then the matrix $\mathrm{Syl}_4(G_0, H_0, x)$ is as follows.

$$\mathrm{Syl}_4(G_0, H_0, x) = \begin{pmatrix} u_1^2 u_2 & u_1 u_2 & u_1 - u_2 & 0 \\ 0 & u_1^2 u_2 & u_1 u_2 & u_1 - u_2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The resultant of G_0 and H_0 w.r.t. x is $u_1^2 - 2u_1 u_2 + u_2^2$, and the set $\{(A_i, B_i)\}_{i=0}^{3}$ are shown in the following table.

Table 1: The set $\{(A_i, B_i)\}_{i=0}^{3}$ of $F(x, u_1, u_2)$ on Example 3.3

i	A_i	B_i
0	$-x u_1 u_2 + u_1 - u_2$	$x u_1^3 u_2^2 - u_1^3 u_2 + 2u_1^2 u_2^2$
1	$x u_1 - x u_2$	$-x u_1^3 u_2 + x u_1^2 u_2^2 - u_1^2 u_2 + u_1 u_2^2$
2	0	$u_1^2 - 2u_1 u_2 + u_2^2$
3	0	$x u_1^2 - 2x u_1 u_2 + x u_2^2$

$k = 1$: $\delta F^{(1)} = x^3 u_1 u_2^2$. Since $H_0 \mid \delta F^{(1)}$, we have $\delta H^{(1)} = 0$ and $\delta G^{(1)} = x u_1 u_2^2$. The Hensel factors of F are: $G^{(1)} = x^2 u_1^2 u_2 + x u_1 u_2 + u_1 - u_2 + x u_1 u_2^2$, $H^{(1)} = x^2$.

$k = 2$: $\delta F^{(2)} = 3x u_1 u_2$. We introduce the auxiliary variable $\mathscr{M}$ to replace $\frac{1}{u_1^2 - 2u_1 u_2 + u_2^2}$. Let $c_0^{(2)} = 0$ and $c_1^{(2)} = 3u_1 u_2$, then $\delta H^{(2)} = \mathscr{M}(\sum_{i=0}^{1} c_i^{(2)} A_i) = 3\mathscr{M} x u_1 u_2 (u_1 - u_2)$ and

$\delta G^{(2)} = \mathscr{M}(\sum_{i=0}^{1} c_i^{(2)} B_i) = 3\mathscr{M} u_1u_2(-xu_1^3u_2 + xu_1^2u_2^2 - u_1^2u_2 + u_1u_2^2)$. The Hensel factors of F are:

$$\begin{cases} G^{(2)} = G^{(1)} + \frac{3u_1u_2(-xu_1^3u_2+xu_1^2u_2^2-u_1^2u_2+u_1u_2^2)}{u_1^2-2u_1u_2+u_2^2}, \\ H^{(2)} = H^{(1)} + \frac{3xu_1u_2(u_1-u_2)}{u_1^2-2u_1u_2+u_2^2}. \end{cases}$$

For $k = 3 \to 4 \to \cdots$, repeat Step 4.1 and Step 4.2.

3.3 A Criterion to Improve the New Algorithm

Under the assumption that the set $\{(A_i, B_i)\}_{i=0}^{n+l-1}$ has been calculated, $\mathcal{G}^{(k)}$ and $\mathcal{H}^{(k)}$ can be quickly solved by a number of multiplication and addition as described in section 3.1. Therefore, the most critical part of our algorithm is the computation of A_i and B_i in an efficient manner. In the following, we list a criterion to quickly calculate the set $\{(A_i, B_i)\}_{i=0}^{n+l-1}$.

Criterion: Calculate (A_{i+1}, B_{i+1}) by using the results of (A_i, B_i), where $0 \le i \le n+l-2$.

Suppose that (A_i, B_i) has been calculated. Multiplying both sides of Equation (9) by x, we get

$$x^{i+1} \cdot \mathrm{res}_x(G_0, H_0) = (x \cdot A_i)G_0 + (x \cdot B_i)H_0. \tag{14}$$

If $\deg_x(x \cdot A_i) < \deg_x(H_0)$ and $\deg_x(x \cdot B_i) < \deg_x(G_0)$, then $(A_{i+1}, B_{i+1}) = (x \cdot A_i, x \cdot B_i)$. If these degree constraints are not satisfied, we must reduce the degrees of $x \cdot A_i$ and $x \cdot B_i$ w.r.t. x. For the degree reduction, the next proposition is very useful.

Proposition 3.4 *If* $\deg_x(x \cdot A_i) \ge \deg_x(H_0)$ *and* $\deg_x(x \cdot B_i) \ge \deg_x(G_0)$, *then* $A_{i+1} = \mathrm{rem}(x \cdot A_i, H_0)$ *and* $B_{i+1} = \mathrm{rem}(x \cdot B_i, G_0)$.

Proof Since $x^{i+1} \cdot \mathrm{res}_x(G_0, H_0)$ can be expressed in two ways, we obtain $(A_{i+1} - x \cdot A_i)G_0 = -(B_{i+1} - x \cdot B_i)H_0$. Because G_0 and H_0 are relatively prime, it follows that $G_0 \mid (B_{i+1} - x \cdot B_i)$ and $H_0 \mid (A_{i+1} - x \cdot A_i)$. These relations and degree inequalities tell us that $A_{i+1} = \mathrm{rem}(x \cdot A_i, H_0)$ and $B_{i+1} = \mathrm{rem}(x \cdot B_i, G_0)$. ▮

Remark 3.5 The **Criterion** implies that if we know the values of (A_0, B_0), then (A_i, B_i) $(1 \le i \le n+l-1)$ can be quickly calculated. It follows from Proposition 3.1 that we can get $\mathrm{res}_x(G_0, H_0)$, A_0 and B_0. There are many ways such as sparse interpolation [5] to compute its resultant. In the process of calculation, there is a critical step. If $\mathrm{res}_x(G_0, H_0)$, A_0 and B_0 have been obtained, we will first compute their greatest common divisor $d = \gcd(\mathrm{res}_x(G_0, H_0), A_0, B_0)$, then $\mathrm{res}_x(G_0, H_0) \triangleq \frac{\mathrm{res}_x(G_0,H_0)}{d}$, $A_0 \triangleq \frac{A_0}{d}$ and $B_0 \triangleq \frac{B_0}{d}$. This step can help us improve computational efficiency when we compute $\delta G^{(k)}$ and $\delta H^{(k)}$.

4 Comparison of Three Algorithms

In this section, we first informally sketch the main idea underlying MY and SI algorithms, followed by a comparison between them and our algorithm with the help of three examples.

4.1 MY Algorithm and SI Algorithm

The MY algorithm is based on interpolation functions, and all of the computational operations are performed in $\mathbb{K}(\mathbf{u})[x]$. However, the computation of the interpolation functions with the extended Euclidean algorithm requires a significant amount of processing time. The first important step of the SI algorithm is to compute the syzygies of elements in the Gröbner basis. Second, their method requires the introduction of many different auxiliary variables in order to avoid dealing with fractional functions when polynomials are reduced to nonzero by using Gröbner basis. We refer [6] and [9, 10] for more details of the MY and SI algorithms.

4.2 Implementation and Discussion

Three algorithms are implemented on a computational algebra system named *Magma*, and the computation was done on a computer with Intel(R) Core(TM) i7-4790 CPU(3.60GHz), operated by Windows 7. Next, we use three examples to compare the computational efficiency of the three algorithms. Each datum in the following figure and Tables is an average of 1000 repetitions of corresponding unit operation.

Example 4.1 Let $F_1 = u_1^2u_2x^4 + (u_1^3u_2^2 + u_1u_2^2 + u_1u_2)x^3 + (u_1^4u_2 + u_1^2u_2^3 + 2u_1^2u_2^2 + u_1 - u_2)x^2 + (u_1^3u_2^2 + u_1^3u_2 + u_1^2u_2 + u_1u_2^3 + 3u_1u_2)x + u_1^3 - u_1^2u_2 + u_1u_2 - u_2^2$.

We use three algorithms to calculate the Hensel factors of F_1, and get the following figure.

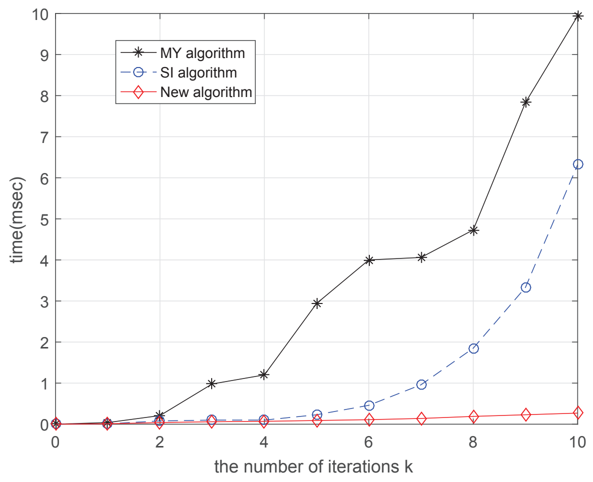

Figure 2 Runtime (msec) of three algorithms on Example 4.1

Example 4.2 Let $F_2 = GH + 3x^{10}u_1^5u_2^5$, where G and H are as follows.

$$\begin{cases} G = (x^5u_1^3 + 2u_2)(x^5u_2^3 - 2u_1) + x^3u_2^4, \\ H = (x^5u_1^3 - 3u_2)(x^5u_2^3 + 3u_1) + x^7u_1^6. \end{cases} \tag{15}$$

F_2 has only one Newton line, and the corresponding Newton polynomial $\overline{F_2}_{\mathcal{N}}(x, u_1, u_2)$ is $(x^5u_1^3+2u_2)(x^5u_2^3-2u_1)(x^5u_1^3-3u_2)(x^5u_2^3+3u_1)$. Without loss of generality, let $\deg_x(G_0) = 10$

and $\deg_x(H_0) = 10$, then we have three choices for the initial Hensel factors G_0 and H_0:

$$\begin{cases} \textbf{C1}: \begin{cases} G_0 = (x^5u_1^3 + 2u_2)(x^5u_2^3 - 2u_1), \\ H_0 = (x^5u_1^3 - 3u_2)(x^5u_2^3 + 3u_1); \end{cases} \\ \textbf{C2}: \begin{cases} G_0 = (x^5u_1^3 - 3u_2)(x^5u_2^3 - 2u_1), \\ H_0 = (x^5u_1^3 + 2u_2)(x^5u_2^3 + 3u_1); \end{cases} \\ \textbf{C3}: \begin{cases} G_0 = (x^5u_1^3 - 3u_2)(x^5u_1^3 + 2u_2), \\ H_0 = (x^5u_2^3 + 3u_1)(x^5u_2^3 - 2u_1). \end{cases} \end{cases} \tag{16}$$

For the three choices mentioned above, we get the following three tables by calculation.

Table 2: timing data (msec) about computing the Hensel factors of F_2 by using New algorithm

Comp.step	**C1**	**C2**	**C3**
$\{(A_i, B_i)\}_{i=0}^{19}$	0.532	0.212	0.472
$G^{(14)}$,$H^{(14)}$	0.110	0.090	0.440
$G^{(16)}$,$H^{(16)}$	0.120	0.130	1.260
$G^{(18)}$,$H^{(18)}$	0.200	0.140	1.640
$G^{(20)}$,$H^{(20)}$	0.450	0.140	2.350

Table 3: timing data (msec) about computing the Hensel factors of F_2 by using MY algorithm

Comp.step	**C1**	**C2**	**C3**
MY-functions	10.660	4.750	9.560
$G^{(14)}$,$H^{(14)}$	1.690	7.660	21.490
$G^{(16)}$,$H^{(16)}$	4.380	10.800	22.470
$G^{(18)}$,$H^{(18)}$	6.720	17.060	44.840
$G^{(20)}$,$H^{(20)}$	14.340	18.340	60.810

Table 4: timing data (msec) about computing the Hensel factors of F_2 by using SI algorithm

Comp.step	**C1**	**C2**	**C3**
Syzygy	0.090	0.070	0.100
$G^{(14)}$,$H^{(14)}$	0.110	0.350	2.860
$G^{(16)}$,$H^{(16)}$	0.200	0.680	6.230
$G^{(18)}$,$H^{(18)}$	0.240	1.090	12.860
$G^{(20)}$,$H^{(20)}$	0.440	1.820	23.400

Example 4.3 Let $F_3 = -x^{40}u_1^{12}u_2^{12} + 2x^{37}u_1^{23}u_2^4 - x^{36}u_1^5u_2^{22} + 2x^{35}u_1^{10}u_2^{17} + 6x^{34}u_1^{28} - 4x^{34}u_1^{15}u_2^{12} + 5x^{33}u_1^{16}u_2^{12} + 3x^{33}u_1^{11}u_2^{16} - 2x^{32}u_1^{12}u_2^{16} + x^{31}u_1^{18}u_2^{10} - x^{30}u_2^{30} - 4x^{25}u_1^{30} - 6x^{25}u_2^{30} + x^{20}u_1^{20}u_2^{12} + x^{20}u_1^{12}u_2^{20} - 24x^{10}u_1^{18}u_2^{18} + 4x^5u_1^{38} + 6x^5u_2^{38} - u_1^{20}u_2^{20}$.

F_3 has only one Newton line of slope: $\lambda_1 = -\frac{2}{5}$, and the corresponding Newton polynomial $\overline{F_3}_{\mathcal{N}}(x, u_1, u_2)$ is $(x^{20}u_2^{12} + 4x^5u_1^{18} - u_2^{20})(x^{20}u_1^{12} + 6x^5u_2^{18} - u_1^{20})$. Without loss of generality, we assume that the initial factors are:

$$\begin{cases} G_0 = x^{20}u_2^{12} + 4x^5u_1^{18} - u_2^{20}, \\ H_0 = x^{20}u_1^{12} + 6x^5u_2^{18} - u_1^{20}. \end{cases} \tag{17}$$

We use three algorithms to calculate the Hensel factors of F_3, and get the following table.

Table 5: timing data (sec) about computing the Hensel factors of F_3 by using three algorithms

Comp.step	New algorithm	MY algorithm	SI algorithm
$\{(A_i, B_i)\}_{i=0}^{39}$	2.760		
MY-functions		1.650	
Syzygy			0.080
$G^{(1)}$,$H^{(1)}$	0.001	0.020	0.030
$G^{(2)}$,$H^{(2)}$	0.010	0.670	0.170
$G^{(3)}$,$H^{(3)}$	0.060	3.670	0.720
$G^{(4)}$,$H^{(4)}$	0.580	18.800	2.960
$G^{(5)}$,$H^{(5)}$	4.180	93.190	9.620
$G^{(6)}$,$H^{(6)}$	20.620	218.440	30.150

We can obtain the following conclusions from the timing data in Example 4.1, Example 4.2 and Example 4.3:

(1). The computational efficiency of our new algorithm is better than that of the MY algorithm and SI algorithm.

(2). There exists a major fault in the MY algorithm: handling polynomials with coefficients in $\mathbb{K}(\mathbf{u})$ is time-consuming.

(3). In order to make $\delta H^{(k)}$ and $\delta G^{(k)}$ satisfy the constraint conditions, the SI algorithm requires a degree reduction w.r.t. x during each iterative process. Moreover, the number of auxiliary variables may increase a lot, which leads to the calculation of Hensel factors being particularly troublesome.

(4). In Example 4.2, different initial factors have great influence on the calculation of EHC.

(5). In Example 4.3, our new algorithm takes a lot of time to compute the coefficients of $\delta F^{(k)}$ w.r.t. x.

5 Conclusions

We have studied the EHC of multivariate polynomials. The first conclusion to be drawn from the experimental data presented earlier is that our algorithm, which is based on the resultant, is more efficient for computing the EHC of multivariate polynomials.

A point that needs to be stressed is that an important step in the SI algorithm is based on the calculation of a polynomial $\widehat{g} \in \mathbb{K}[\mathbf{u}]$ such that it is a factor of $\mathrm{res}_x(G_0, H_0)$. Sasaki and Inaba compute $\widehat{g}$ by using the Gröbner basis [9], and also propose some enhancements to improve their algorithm in [10]. Next we compare and analyze the New algorithm and the SI algorithm to calculate $\mathrm{res}_x(G_0, H_0)$ and $\widehat{g}$ respectively, when we increase the number of sub-variables.

We assume that G_0 and H_0 are high order sparse polynomials. We randomly generate three groups of polynomials, each group having 10 polynomials that are relatively prime. In addition, the number of sub-variables of the three groups is between 2 and 4. Each polynomial satisfies the following conditions: (1). the degree w.r.t. x is at least 15; (2). the number of total terms is at most 4. Each datum in Table 6 is an average of the total time of each group.

Table 6: timing data (sec) about computing $\mathrm{res}_x(G_0, H_0)$ and $\widehat{g}$ respectively

sub-variables	New algorithm	SI algorithm
2	0.800	0.976
3	2.265	18.030
4	7.135	69.440

From Table 6 we can draw the following conclusion: As the number of sub-variables increases, the computation of the Gröbner basis becomes more and more complex, but it has little effect on the calculation of resultant. Consequently, our algorithm is faster than the SI algorithm.

Finally, we think that more work needs to be done in regard to the efficient calculation of the resultant of G_0 and H_0 w.r.t. x. Experiments show that the computational efficiency of resultant is becoming slower and slower as the degree of G_0 and H_0 w.r.t. x increases. In the case where G_0 and H_0 are all sparse polynomials, new ideas are needed to utilize the sparseness of polynomials.

Acknowledgements The authors would like to thank anonymous referees for detailed suggestions on the paper which have made it more readable.

References

[1] Cox D, Little J and O'Shea D, *Ideals, Varieties, and Algorithms*, Springer-Verlag, New York, 2007.

[2] Geddes K, Czapor S and Labahn G, *Algorithms for Computer Algebra*, Springer US, 1992.

[3] Inaba D, Factorization of multivariate polynomials by extended Hensel construction, *ACM SIGSAM Bulletin*, 2005, **39(1)**:2–14.

[4] Kleine J, Monagan M and Wittkopf A, Algorithms for the non-monic case of the sparse modular GCD algorithm, *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, Beijing, 2005.

[5] Monagan M and Tuncer B, Using sparse interpolation in Hensel lifting, *Proceedings of the 18th International Workshop on Computer Algebra in Scientific Computing*, Romania, 2016.

[6] Moses J and Yun D, The EZ GCD algorithm, *Proceedings of the ACM Annual Conference*, Atlanta, Georgia, USA, 1973.

[7] Musser D, Algorithms for polynomial factorization, Doctoral Dissertation, the University of Wisconsin, Madison, 1971.

[8] Sanuki M, Inaba D and Sasaki T, Computation of GCD of sparse multivariate polynomials by extended Hensel construction, *Proceedings of the 17th Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Romania, 2015.

[9] Sasaki T and Inaba D, Enhancing the extended Hensel construction by using Gröbner bases, *Proceedings of the 18th International Workshop on Computer Algebra in Scientific Computing*, Romania, 2016.

[10] Sasaki T and Inaba D, Various enhancements for extended Hensel construction of sparse multivariate polynomials, *Proceedings of the 18th Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Romania, 2016.

[11] Sasaki T and Kako F, Solving multivariate algebraic equation by Hensel construction, *Japan Journal of Industrial and Applied Mathematics*, 1999, **16(2)**:257–285.

[12] Sasaki T and Sasaki M, A unified method for multivariate polynomial factorizations, *Japan Journal of Industrial and Applied Mathematics*, 1993, **10**:21–39.

[13] Wang P and Rothschild L, Factoring multivariate polynomials over the integers, *ACM SIGSAM Bulletin*, 1973, **28**:21–29.

[14] Zippel R, Probabilistic algorithms for sparse polynomials, *Proceedings of the International Symposiumon Symbolic and Algebraic Computation*, France, 1979.