Approximate GCD of Multivariate Polynomials

L.H. Zhi, M.-T. Noda
Dept. of Computer Science, Ehime University,
Matsuyama 790-8577, Japan
E-mail:{lzhi,noda}@hpc.cs.ehime-u.ac.jp

We describe algorithms for computing the greatest common divisor of two multivariate polynomials with inexactly known coefficients. We focus on extending standard exact EZ-GCD algorithm to an efficient and stable algorithm in approximate case. Various issues related to the implementation of the algorithms and some preliminary test results are also presented.

1 Introduction

Given two polynomials F and G in $R[x_1, \ldots, x_n]$, we are going to find the nontrivial approximate GCD C and polynomials $F', G' \in R[x_1, \ldots, x_n]$ such that $||F - CF'|| < \epsilon$ and $||G - CG'|| < \epsilon$, for some ϵ and some well defined norm. Many papers [1,2,3,5,8,10,11,13,15] have already discussed the problem in the case $n = 1$. Few of them [2,10,11] mentioned the case $n > 1$. Approximate GCD computation of univariate polynomials provides the basis for solving multivariate problem. But it is nontrivial to modify the techniques used in symbolic computation such as interpolation and Hensel lifting to compute the approximate GCD of multivariate polynomials.

In section 2, we briefly review two methods [2,10] for computing GCD of polynomials with floating-point coefficients. In section 3, we focus on extending the Hensel lifting technique to polynomials with floating-point coefficients. The method involves the QR decomposition of a Sylvester matrix. We propose an efficient new algorithm which exploits the special structure of Sylvester matrix. A local optimization problem is also been proposed to improve the candidate approximate factors obtained from Hensel lifting.

In order to compare the performance of the different methods, we implement all three methods in Maple. In section 4, we summarize the special problems we encounter when they are applied to polynomials with floating-point coefficients. A set of examples are given to show the efficiency and stability of the algorithms.

2 Review of GCD algorithms

Modular Algorithm: Multivariate polynomial GCD problems are reduced to univariate problems by using(repeatedly) evaluation homomorphisms to elim-

inate variables, and reconstruct the GCD of the original inputs from these "images" using interpolations. For the dense modular algorithm, the number of homomorphisms is exponential in the number of variables. If the multivariate polynomials and their non-trivial GCD are sparse, then sparse modular method needs much less number of homomorphisms. Corless et al. [2] modified the sparse modular algorithm to compute approximate GCD in the case of bivariate polynomials.

1. Choose random evaluations α, β_i of x and y to compute T_x nonzero terms in $\mathrm{GCD}(F(x,\beta_i), G(x,\beta_i))$ and T_y nonzero terms in $\mathrm{GCD}(F(\alpha,y), G(\alpha,y))$.
2. Solve M monic $\mathrm{GCD}(F(x,\beta_i), G(x,\beta_i))$ for randomly chosen β_i, where $M \geq T_y T_x/(T_x - 1)$.
3. Interpolate all coefficients simultaneously.

Subresultant PRS: The Subresultant PRS algorithm calculates the PRS

$$\{P_1 = F, P_2 = G, \ldots, P_k \neq 0, P_{k+1} = 0.\} \tag{1}$$

by the iteration formula

$$\beta_i P_{i+1} = \mathrm{remainder}(\alpha_i P_{i-1}, P_i), i = 2, 3, \ldots. \tag{2}$$

where,

$$\begin{array}{l} \alpha_i = \mathrm{lc}(P_i)^{d_i+1}, \quad d_i = \deg(P_{i-1}) - \deg(P_i); \\ \beta_2 = 1, \quad \gamma_2 = 1; \\ \beta_i = \mathrm{lc}(P_{i-1}), \quad \gamma_i = \mathrm{lc}(P_{i-1})^{d_i-1} \gamma_{i-1}^{1-d_{i-1}}, i \geq 3. \end{array} \tag{3}$$

$\mathrm{lc}(P)$ denotes the leading coefficient of P. The above algorithm has been extended to polynomials with floating-point coefficients [10]. There are two critical steps:

1. Normalization of remainders in PRS. Let R and Q be pseudo-remainder and pseudo-quotient of polynomials F and G, i.e.,

$$R = \mathrm{lc}(G)^{d+1} F - QG, \quad d = \deg(F) - \deg(G). \tag{4}$$

   The normalized pseudo-remainder $\widetilde{R}$ is defined as

$$\widetilde{R} = \frac{R}{\max(\| \mathrm{lc}(G)^{d+1} \|, \| Q \|)}. \tag{5}$$

2. Rounding of P_k. Normalize P_k as $\| P_k \| = 1$, where $\| \cdot \|$ denotes the maximum norm of the coefficient vector. Round off the coefficients of P_k at $\mathsf{O}(\varepsilon)$.

3 Hensel Lifting Algorithm

For polynomials with exact coefficients, the main steps compute GCD using Hensel lifting are summarized below [4,14].

1. Choose a main variable, suppose x_1, find lucky evaluation homomorphism $I = (x_2 - a_2, \ldots, x_n - a_n)$.
2. $F_I = F \mod I, G_I = G \mod I$. Compute $C_I = \mathrm{GCD}(F_I, G_I)$ and cofactors $\widetilde{F_I}, \widetilde{G_I}$.
3. If there is one of the cofactors which is prime to C_I, then use multivariate Hensel construction to lift C_I and the cofactor. Otherwise perform a square-free decomposition for either F or G.

The last step is critical. Suppose P is a polynomial in $x_1, \ldots, x_n$ with leading coefficient $p_m(a_2, \ldots, a_n) \neq 0$. Let $x = x_1, \mathsf{u} = \{x_2, \ldots, x_n\}$, $G^{(0)}(x)$ and $H^{(0)}(x)$ be relatively prime polynomials satisfying

$$P(x, \mathsf{u}) \equiv G^{(0)}(x)H^{(0)}(x) \mod I. \tag{6}$$

The multivariate Hensel construction calculates polynomials $G^{(k)}(x, \mathsf{u})$ and $H^{(k)}(x, \mathsf{u})$ satisfying

$$P(x, \mathsf{u}) \equiv G^{(k)}(x, \mathsf{u})H^{(k)}(x, \mathsf{u}) \mod I^{k+1}. \tag{7}$$

The main calculation involved in multivariate Hensel construction is to solve polynomial Diophantine equations for the fixed polynomials $G^{(0)}(x)$ and $H^{(0)}(x)$. If $G^{(0)}(x)$ and $H^{(0)}(x)$ are relatively prime, then for any polynomial $R(x)$ with $\deg(R(x)) < \deg(G^{(0)}(x)) + \deg(H^{(0)}(x))$, there exist unique polynomials $A(x), B(x)$ such that

$$A(x)G^{(0)}(x) + B(x)H^{(0)}(x) = R(x) \tag{8}$$

and

$$\deg(A(x)) < \deg(H^{(0)}(x)), \quad \deg(B(x)) < \deg(G^{(0)}(x)).$$

Solving (8) is equivalent to solving the linear equations with fixed coefficient matrix M, where M is Sylvester matrix of polynomial $G^{(0)}(x)$ and $H^{(0)}(x)$. There are several ways to solve the linear equations. LU decomposition is the fastest but may be unstable in some cases, especially when M is close to singular; SVD method can detect nearly rank-deficiency in the presence of roundoff error and give satisfactory results while LU decomposition fails, but it is notoriously computational-intensive; QR decomposition with Householder

transformations or Given rotations is very stable although it costs double time than LU decomposition. We prefer QR decomposition as it is easy to exploit the structure of the Sylvester matrix M for the purpose of finding a computationally efficient and stable algorithm.

Since the Sylvester matrix M consists of two Toeplitz matrices, we start the QR decomposition with the Given rotation and to eliminate the elements below the diagonal of the first t columns, and then apply the Householder transformations to the low $s \times s$ submatrix (suppose $s >= t$).

The advantage of combining Given rotations with Householder transformations can be seen clearly from comparing complexity in the case $s = t = n$. The flops used in general LU, QR and SVD decomposition are $\frac{2}{3}(2n)^3$, $\frac{4}{3}(2n)^3$ and $12(2n)^3$; while using the above strategy, the flops we used are $6n^2 + \frac{4}{3}n^3$.

For Hensel construction, it is also very important to decide when to stop the lifting. Obviously, the procedure will stop as soon as $\| \Delta P^{(k)} \| = \| P - G^{(k)}H^{(k)} \| = \mathsf{O}(\varepsilon)$. However, it is also possible that $\Delta P^{(k)}$ still has some coefficients whose absolute values are not $\mathsf{O}(\varepsilon)$ when k is larger than the total degree of the variables u in P. In this case, we have to decide if it is caused by error accumulation or P has no approximate factors indeed. If many coefficients of $\Delta P^{(k)}$ are relatively much larger than ε, it is believed that F and G have no approximate GCD. Otherwise, we can use the optimization method [11] to improve the initial approximate univariate GCD and its cofactor $G^{(0)}$ and $H^{(0)}$, or increase the number of digits carried in floating-point computations. But it is possible that both techniques are inefficient. Then we have to apply the correction method that was given in a separate paper by author and others [6].

When $\Delta P^{(k)}$ is reasonable small, we round off $G^{(k)}$ and $H^{(k)}$ to G and H respectively. G and H are supposed to be candidate factors of P. If $\| P - GH \|$ is not very large, we formulate a local minimization problem:

$$\min_{\Delta G, \Delta H} \| P - GH - G\Delta H - \Delta G H \| . \tag{9}$$

Notice that it is not necessary to actually solve the minimization problem. We are only interested in finding $\Delta H, \Delta G$ which modify H and G to achieve a sufficient small backward error.

Example 1:

$$\begin{aligned} F &= (x^2 + y^2 + 1.01)(x^2 + xy + y^2 + 1.12), \\ G &= (x^2 + y^2 + 1.01)(x^2 + xy + 1.02) + 10^{-4}(x + y). \end{aligned} \tag{10}$$

For $\varepsilon = 10^{-4}$, $y = 0$, we modify algorithms in the papers [10,11] to get

$$C^{(0)} = \mathrm{GCD}(F(x,0), G(x,0)) = 1.01 - 0.000998478x + 1.x^2,$$

with backward error about 10^{-5} and cofactor of $C^{(0)}$ with respect to $F(x,0)$

$$\widetilde{F}^{(0)} = 1.12 + 0.001055x + 1.x^2.$$

For simplicity, polynomials here and below are rounded to six digits.

Compute the Hensel lifting for $F, C^{(0)}$ and $\widetilde{F}^{(0)}$. When $k = 3$,

$$\begin{aligned} C^{(2)} &= 1.01 - 0.000998478x + 1.x^2 + 0.99999y^2, \\ \widetilde{F}^{(2)} &= 1.12 + 0.001055x + 1.x^2 + 0.999821xy + 0.0101626y + 1.00001y^2, \\ &\| F - C^{(2)}\widetilde{F}^{(2)} \| = 0.0102642. \end{aligned}$$

Since the leading coefficients of F, $C^{(2)}$ and $\widetilde{F}^{(2)}$ with respect to x are equal to 1, no corrections are needed. The optimization problem (9) is the minimization of 14 linear equations in 8 correction coefficients subject to 6 linear equality conditions. It turns out we can choose all corrections be zero except the correction coefficient of y in $\widetilde{F}^{(2)}$ be -0.01. This gives us a backward error about 10^{-3}.

For this example, it is very difficult to get the correct approximate GCD only by improving the univariate GCD or increasing the Digits in Maple (e.g. Digits=40). We will discuss this problem in section 4.

4 Implementation and Discussion

It has been pointed out [9] that all these three methods have advantages and disadvantages. For example: sparse modular method can find the "small" GCD very fast but may be unlucky or need a large number of evaluation homomorphisms in the case the GCD is dense; Subresultant algorithm can deal with the polynomials with complicated leading coefficients but it is very inefficient to find that two polynomials are actually prime to each other; Hensel lifting algorithm only needs small number of homomorphisms but has trouble if the GCD is not prime to any of its cofactor. All these advantages and disadvantages are remained and some special drawback will appear when we extend these methods to polynomials with floating-point coefficients.

4.1 Modular Algorithm

The main serious problem in Modular Algorithm is the high condition number of the Vandermonde matrix for interpolation of polynomials. A variety of bounds for the condition number of a Vandermonde matrix have been derived by Gautschi [7]. Let $V_n = V(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{R}^{n\times n}$.

Corless et al. suggest to randomly scale each variable and choose the random points: $\alpha + \beta u_i$, where u_i are the Chebyshev nodes in $[-1, 1]$ and

Table 1: Bounds and Estimates for $k_\infty(V_n)$.

α_i	$k_\infty(V_n)$	$n=5$	$n=6$	$n=7$
equispaced $[0,1]$	$(4\pi)^{-1}\sqrt{2}\,8^n$	3687.7	29501.5	236012.5
equispaced $[-1,1]$	$\pi^{-1}\varepsilon^{-\pi/4}(3.1)^n$	41.5	128.8	399.3
Chebyshev nodes $[-1,1]$	$\frac{3^{3/4}}{4}(1+\sqrt{2})^n$	46.7	112.8	272.4
$\alpha_i \geq 0$	$> 2^{n-1}(n>1)$	$10^3 \sim 10^4$	$10^4 \sim 10^5$	$10^5 \sim 10^6$

$\alpha, \beta \in (1/2, 1)$. It is still possible to get very large condition numbers when n is big. We have tried to use orthogonal polynomials instead of power series $1, x, x^2, \ldots,$ to interpolate the GCDs. The details will be reported in another paper. However, it is always very dangerous to interpolate a polynomial from more than 10 points [12]. So it is not recommended to interpolate a non-monic GCD with degree larger than 5 and dense in some variables.

In Corless's paper [2], in order to obtain the same denominator of the GCD, a large system is formed to interpolate all coefficients simultaneously. However, it is possible to interpolate coefficients independently if we perform the normalization of the leading coefficient [4]. Firstly, we estimate the leading coefficient of the GCD from the relation

$$\mathrm{lc}(\mathrm{GCD}(F,G)) \,|\, \mathrm{GCD}(\mathrm{lc}(F), \mathrm{lc}(G)). \tag{11}$$

If $c = \mathrm{GCD}(\mathrm{lc}(F), \mathrm{lc}(G))$ is small, we normalize the GCD to have the leading coefficient c. Now the coefficients of the GCDs we computed will be polynomials in all variables and can be interpolated one by one. However, if we are unlucky, the GCD we computed may contain an extra factor which can be divided out after computing the content of the GCD(it requires to run the algorithm recursively). Moreover, we also notice that normalization of the leading coefficient may destroy the sparsity of the GCD. So normalization is only practical when c is simple (e.g. 1).

Sparse Modular Algorithm is fast but also probabilistic. The algorithm is not efficient for dense GCDs and could also be unlucky. In floating-point computations, "zero" only means relatively small, i.e., $\mathsf{O}(\varepsilon)$. Failure to recognize the essential "zero" will bring unsatisfactory answer.

Example 2:

$$\begin{aligned} F &= (1+x+y_1+y_2+y_3+y_4+y_5)^2 \\ &\quad (-2+x-(y_1+y_2+y_3+y_4+y_5)^2), \\ G &= (1+x+y_1+y_2+y_3+y_4+y_5)^2 \\ &\quad (2+x+y_1+y_2+y_3+y_4+y_5)^2. \end{aligned} \tag{12}$$

Compute $T_x = T_{y_1} = T_{y_2} = T_{y_3} = T_{y_4} = T_{y_5} = 3$. Since the GCD is monic

and dense, we need about $3^5 = 243$ evaluations. For Digits=10, $\varepsilon = 10^{-5}$, the algorithm returns:

$$
\begin{aligned}
&2.00029x + 1.x^2 + 1.99931y_1x + 1.99909y_1 + 1.99789y_2 + 1.99773y_3 \\
&+1.99476y_4 + 1.99543y_5 + 1.00134y_1^2 + 1.00188y_2^2 + 1.00202y_3^2 + 1.00512y_4^2 \\
&+1.00550y_5^2 - 0.112238y_2y_4y_5 - 0.0681706y_2y_3y_5 - 0.120297y_3y_4y_5 \\
&-0.0525764y_1y_4y_5 - 0.0387045y_1y_2y_5 - 0.0263314y_1y_3y_5 + 0.0835444y_5^2y_2y_3 \\
&+0.0199011xy_4y_5 + 0.0640616y_5^2y_1y_4 + 0.135066y_5^2y_2y_4 + 0.1476623y_5^2y_3y_4 \\
&+0.0461392y_5^2y_1y_2 + 0.0353559y_5^2y_1y_3 - 0.0237874y_5^2xy_4 + 0.0197683y_3^2y_5^2 \\
&+0.0466600y_4^2y_5^2 + 0.0125651y_1^2y_5^2 + 0.0141069y_2^2y_5^2 - 0.0480776y_4y_5^2 \\
&-0.0102278y_5y_1^2 - 0.0125707y_5y_2^2 - 0.0158431y_5y_3^2 - 0.0387656y_5y_4^2 \\
&-0.0181389y_2y_5^2 - 0.0218871y_3y_5^2 + 2.03987y_4y_5 + 1.99737xy_4 \\
&+1.99786xy_5 + 2.00681y_1y_4 + 2.00683y_1y_5 + 2.01477y_2y_4 + 2.01531y_2y_5 \\
&+2.01526y_3y_4 + 2.01765y_3y_5 + 1.00060 + 1.99889xy_2 \\
&+1.99894xy_3 + 2.00545y_1y_2 + 2.00329y_1y_3 + 2.00916y_2y_3.
\end{aligned}
$$

Some terms such as $y_2y_4y_5, \ldots, y_2y_3y_5^2$ which should be zeros but have not very small coefficients here. If Digits=20, we can get the exact result:

$$
\begin{aligned}
&2.x + 1. + 1.x^2 + 2.y_1x + 2.y_1 + 2.y_2 + 2.y_3 + 2.y_4 + 2.y_5 + 1.y_1^2 + 1.y_2^2 + 1.y_3^2 \\
&+1.y_4^2 + 1.y_5^2 + 2.y_4y_5 + 2.xy_4 + 2.xy_5 + 2.y_1y_4 + 2.y_1y_5 + 2.y_2y_4 + 2.y_2y_5 \\
&+2.y_3y_4 + 2.y_3y_5 + 2.xy_2 + 2.xy_3 + 2.y_1y_2 + 2.y_1y_3 + 2.y_2y_3.
\end{aligned}
$$

4.2 Subresultant PRS

What will happen when we extend Subresultant PRS for polynomials with floating-point coefficients? At each iteration of the algorithm, the pseudo-remainder is divided by a factor which is an exact factor in the case of exact rational coefficients. But now, it is only an approximate factor. Error accumulation is inevitable for each division. Another problem is caused by the large cancelation error during the pseudo-remainder computations. Even we perform the normalization [10], it is still possible to obtain unsatisfactory answers.

Example 3:

$$
\begin{aligned}
F &= y^4 + y + x + 1, \\
G &= y^3 - \eta y + x^2.
\end{aligned}
\tag{13}
$$

Let $\eta = 10^{-3}$ and $\varepsilon = 10^{-6}$, by the approximate Subresultant Algorithm, we get $||P_5|| = \mathsf{O}(\varepsilon)$ and

$$P_4 = (-0.995974x + 0.996977 - 1.00201x^2 + x^3)y - 10.x^2 + 0.997984 + 0.002012x.$$

But the norms of the remainders of F, G w.r.t. P_4 are $\mathsf{O}(1)$.

y	cond(M)	error1	error2	error 3
0.425	7.02	3.41 10^{-5}	8.66 10^{-5}	1.45 10^{-3}
0.65	28.9	5.16 10^{-5}	3.81 10^{-5}	0.15

4.3 Hensel Lifting Algorithm

Now let us see the main difficulty we encountered when apply Hensel lifting to compute GCD of multivariate polynomials with floating-point coefficients. Use the same notations as in section 3, it is clear that if $G^{(0)}(x)$ and $H^{(0)}(x)$ have a close common root, the matrix M will be very ill-conditioned and the multivariate Hensel construction will be sensitive [14]. This may be caused by a bad evaluation or both F and G are not squarefree. For the example 1, the four roots of $G(x,0)$ form two clusters near $\sqrt{-1}$. The roots of $F(x,0)$ are also not very well separated. The condition number of the Sylvester matrix formed by $C^{(0)}$ and $\widetilde{F}^{(0)}$ is about 42. Since the polynomials F and G are of digits 5, it is nature that we obtain an approximate GCD($C^{(2)}$) with backward error 10^{-2} by only performing Hensel lifting.

In order to keep the sparsity and error tolerance of the polynomials, we prefer to choose as many a_i be zero as possible. However, if such a_i leads to bad homomorphism, then we have to choose nonzero evaluation for a_i. It affects not only the efficiency but also the stability of the algorithm. As for the following example:

Example 4:

$$\begin{aligned} F &= 0.25 - 0.25x^2 - 0.25y^2 - 0.9999xy + xy^3 + yx^3 \\ &= (x^2+y^2-1)(xy-0.25) - 10^{-5}xy, \\ G &= -0.00001 + y - 1.00001x + xy^2 - yx^2 + x^3 - y^3 \\ &= (x^2+y^2-1)(x-y) - 10^{-5}(x+1). \end{aligned} \tag{14}$$

We choose x as the main variable. Since the leading coefficient of F with respect to x is zero if we evaluate it at $y = 0$. It is necessary to choose nonzero evaluation for y. For the random values of y belong to [0.5, 1], we can not obtain the meaningful result only by Hensel lifting until Digits = 30; If we choose y in (0, 0.5), then we can get a reasonable result that almost needs no improvement when Digits = 10. The following table compares the performance of the algorithm for example 4 at two different evaluation points. Here, cond(M) is the condition number of M; error1, error2 and error3 correspond to $\| F_I - C_I\widetilde{F}_I \|$, $\| G_I - C_I\widetilde{G}_I \|$ and $\| \Delta P^{(3)} \|$ (lift the two factors of G) respectively.

At last we would like to point out that both Modular Algorithm and Hensel

Lifting Algorithm may be inefficient when they are used to compute approximate non-monic multivariate GCD. On the contrary, Subresultant Algorithm can deal it very well.

Example 5:

$$\begin{aligned} F &= (-1 + xy_1y_2y_3y_4y_5y_6)(3 + xy_1y_2y_3y_4y_4y_5y_6) + 10^{-5}(x + y_1 + y_6), \\ G &= (-1 + xy_1y_2y_3y_4y_5y_6)(-3 + xy_1y_2y_3y_4y_5y_6) + 10^{-5}(y_1 - y_3^2). \end{aligned} \tag{15}$$

Our implementation of Hensel lifting algorithm in Maple fails to obtain the approximate GCD because any evaluation of y_i must be chosen as nonzero which seriously destroy the sparsity of the original polynomials. Much worse, normalization of the leading coefficient introduces a nontrivial extra factor. Modular algorithm takes more than 30 seconds to get a reasonable answer. However, it is trivial using Subresultant Algorithm to obtain approximate GCD:

$$-1. + 1.xy_1y_2y_3y_4y_5y_6 + 0.00001(x + y_3 + y_6). \tag{16}$$

5 Remarks

In this paper, we briefly discuss the three possible methods for computing approximate GCD of multivariate polynomials. Many problems remain open. Especially, we still do not have well developed theory for choosing good homomorphisms. Only some strategies are proposed. It will be the main direction for our future research.

Since both Modular and Hensel Algorithm are based on the approximate univariate GCD computation, we have to choose an efficient and stable algorithm for computing univariate GCDs. Especially for Hensel lifting method, the computation of cofactor of the GCD is also necessary. In our algorithm, we modify algorithm in the paper [10] and use the result as an initial guess, then apply some optimization strategies [11] to improve it.

It is difficult to propose an optimal algorithm for computing GCD of multivariate polynomials with floating-point coefficients and there are also a lot of work to stabilize the algorithms and analyze the errors. Our implementation in Maple has not yet been optimized, therefore we only get some very initial observation from our experiments.

References

1. B. Beckermann and G. Labahn, When are two numerical polynomials relatively prime? *J. Symb. Comput.,* **26**(1998), 677-689.

2. R.M. Corless, P.M. Gianni, B.M. Trager, and S.M. Watt, The singular value decomposition for polynomial systems, *Proc. ISSAC '95*, ACM Press, New York, 1995, 195–207.
3. I.Z. Emiris, A. Galligo and H. Lombaradi, Certified approximate univariate GCDs, *J. Pure and Applied Algebra*, **117** (1997), 229-251.
4. K.O. Geddes, S.R. Czapor and G. Labahn, Algorithms for Computer Algebra, Boston, Kluwer, 1992.
5. V. Hribernig and H.J. Stetter, Detection and validation of clusters of polynomial zeros, *J. Symb. Comput.,* **24**(1997), 667-681.
6. Y. Huang, H.J. Stetter, W. Wu and L.H. Zhi, Pseudofactors of multivariate polynomials, accepted by ISSAC'00.
7. W. Gautschi, How (un)stable are Vandermonde systems? *Lecture Notes in Pure and Applied Mathematics,* **124**(1990), 193-210.
8. N. Karmarkar and Y.N. Lakshman, On approximate polynomial greatest common divisors, *J. Symb. Comp.,***26(6)** (1998), 653-666.
9. H.C. Liao and R.J. Fateman, Evaluation of the heuristic polynomial GCD, *Proc. ISSAC '95*, ACM Press, New York, 1995, 240–247.
10. M. Ochi, M. Noda and T. Sasaki, Approximate greatest common divisor of multivariate polynomials and its application to ill-conditioned system of algebraic equations, *J. Inf. Proces.*, **12**(1991), 292–300.
11. P. Chin, R.M. Corless and G.F. Corliss, Optimization strategies for approximate GCD problem, *Proc. ISSAC '98*, ACM Press, New York, 1998, 228-235.
12. W. Press, B. Flannery, S. Teukolsky and W. Vetterling, Numerical Recipes: The Art of Scientific Computation, Cambridge U. Press, Cambridge, 1990.
13. T. Sasaki and M. Noda, Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations, *J. Inf. Proces.*, **14**(1989), 159-168.
14. T. Sasaki and S. Yamaguchi, An analysis of cancelation error in multivariate Hensel construction with floating-point number arithmetic, *Proc. ISSAC '98*, ACM Press, New York, 1998, 1–8.
15. A. Schönhage, Quasi-GCD computations, *J. Complexity*, **1** (1985), 118–137.