Regular Paper

Approximate Greatest Common Divisor of Multivariate Polynomials and Its Application to Ill-Conditioned Systems of Algebraic Equations

Masa-aki Ochi*, Matu-tarow Noda** and Tateaki Sasaki***†

Let F, $\tilde{F}$ and D be multivariate polynomials and ε be a small positive number, $0<\varepsilon\ll 1$. If $F=D\tilde{F}+\Delta F$, where ΔF is a polynomial with coefficients that are $O(\varepsilon)$-smaller than those of F, D is called an approximate divisor of F of accuracy ε. Given multivariate polynomials F and G, an algorithm is proposed for calculating with accuracy ε the approximate greatest common divisor (GCD) of F and G. The algorithm is a naive extension of the conventional Euclidean algorithm, but it is necessary to treat the polynomials carefully. As an application of the approximate GCD of multivariate polynomials, the solution of a system of algebraic equations $\{F_1(x, y, \cdots, z)=0, \cdots, F_r(x, y, \cdots, z)=0\}$ is considered, where F_i *and* F_j, $i\neq j$, have a non-trivial approximately common divisor. Such a system is ill-conditioned for conventional numerical methods, and is transformed to a well-conditioned system by calculating approximate GCD's. A method is also given for determining the initial approximations of the roots for numerical iterative calculation. The proposed method is tested by using several examples, and the results are very good.

1. Introduction

Recently, two of the present authors (T.S. and M.T.N.) introduced the concept of the approximate greatest common divisor (GCD) of univariate polynomials, constructed a Euclidean algorithm for calculating it, and described an application of the approximate GCD to solving ill-conditioned algebraic equations with one variable [5]. See also Sasaki and Sasaki [6] for a theoretical analysis of the Euclidean algorithm. A concept similar to the approximate GCD, named the "quasi-GCD," was also introduced by Schönhage [8]. The same equation is used to define these concepts, but Schönhage is interested in calculating the "quasi-GCD" quite accurately by using approximate arithmetic; on the other hand, the authors of [5] are interested in considering $(x-\alpha)$ and $(x-\alpha')$ to be approximately equal factors so long as $|\alpha-\alpha'|\leqq\delta$, with δ a given small positive number. The latter need not necessarily be as small such as $d\simeq 10^{-7}$, but may be as large as $\delta\simeq 10^{-2}$ or 10^{-3}. Consequently, close roots whose mutual distance $\leqq\delta$ can be separated as approximately multiple roots by calculating an approximate GCD.

The authors of [5] proposed not only an algorithm for finding the approximate GCD of univariate polynomials, but also an algorithm for approximate square-free decomposition, and pointed out that such approximate calculations could be performed on many algebraic operations. In this paper, we generalize the approximate GCD to multivariate polynomials and show its application to solving a kind of ill-conditioned system of algebraic equations.

The system we will consider is of the kind (for simplicity, we explain the case of two variables)

$$\{F_1(x, y)=0,\ F_2(x, y)=0\}$$

where F_1 and F_2 are polynomials such that

$$F_1(x, y)=D(x, y)\tilde{F}_1(x, y)+\Delta F_1(x, y),$$
$$F_2(x, y)=D(x, y)\tilde{F}_2(x, y)+\Delta F_2(x, y).$$

Here, D is not a constant and ΔF_1 and ΔF_2 are polynomials whose coefficients are very small compared with coefficients of F_1 and F_2. If $\Delta F_1=\Delta F_2=0$, the system $\{F_1=0, F_2=0\}$ has an infinite number of solutions corresponding to the roots of $D=0$, while it usually has a finite number of solutions if $\Delta F_i\neq 0$, $i=1, 2$. This means that the system is ill-conditioned for numerical methods. In fact, the absolute value of the Jacobian becomes quite small around the roots of $D=0$, and the conventional Newton's method requires very many iterations to converge, and may miss some

*Computer Division, Information Equipment Sector, Matsushita Electric Industrial Co., LTD., Kadoma-shi, Osaka 571, Japan.
**Department of Computer Science, Ehime University, Matsuyama-shi, Ehime 790, Japan.
***The Institute of Physical and Chemical Research, Wako-shi, Saitama 351-01, Japan.
†Current address: Institute of Mathematics, University of Tsukuba, Tsukuba-shi, 305, Japan.

roots. In this paper, a system is called ill-conditioned only in the above-mentioned sense.

One way to attack such an ill-conditioned system is to refine Newton's iteration formula, as was done by Tanabe [9]. Another way is to transform the system to an well-conditioned system by using an algebraic method, and the method proposed in this paper is follows this path. As we will show in Section 4, such a transformation can be made easily if the approximate GCD of multivariate polynomials (namely, $D(x, y)$ in the above example) is calculated. Furthermore, knowing the approximate GCD allows us to determine good initial approximations of the roots for a numerical iterative method.

In Section 2, we define the approximate GCD of multivariate polynomials, as well as other necessary notation. A Euclidean algorithm for calculating the approximate GCD is presented in Section 3, where we will see that we must treat polynomials very carefully. An application to the above-mentioned kind of ill-conditioned system of algebraic equations is described in Section 4, and Section 5 gives several examples in which ill-conditioned systems of equation are solved.

2. Approximate GCD of Multivariate Polynomials

The polynomials treated in this paper are included in $C[x, y, \cdots, z]$, where C is the field of complex numbers that may be represented by finite-precision floating-point numbers, and x is treated as the main variable. Let

$$F=f_m x^m+\cdots+f_0,\ f_m\neq 0,$$
$$G=g_n x^n+\cdots+g_0,\ g_n\neq 0,$$

where $f_i\in C[y, \cdots, z]$ and $g_i\in C[y, \cdots, z]$, $i=0, 1, \cdots$. The terms m, f_m, and $f_m x^m$ are the degree, leading coefficient, and leading term, respectively, of F and are denoted as $\deg(F)$, $\mathrm{lc}(F)$, and $\mathrm{lt}(F)$.

Definition 1 [maximum magnitude coefficient]. Let F be a polynomial in $C[x, y, \cdots, z]$. The maximum magnitude numerical coefficient of F is denoted as $\mathrm{mmc}(F)$. //

Definition 2 [numbers of similar magnitude]. Let a and b be numbers in C, with $b\neq 0$. By $a=0(b)$, we mean that $1/c\leqq |a/b|\leqq c$, where c is a positive number close to 1. (Usually, "0" denotes Landau's notation, and we are using "0" in a somewhat different sense.) //

Definition 3 [normalization of polynomial]. Normalization of the polynomial F is the scale transformation $F\rightarrow F'=\eta F$, $\eta\in C$, so that $\mathrm{mmc}(F')=1$. We denote this procedure as $F'=\mathrm{Normalize}(F)$. //

Definition 4 [approximate GCD of accuracy ε]. Let ε be a small positive number, $0<\varepsilon\ll 1$. Let $F(x, y, \cdots, z)$ and $G(x, y, \cdots, z)$ be normalized polynomial in $C[x, y, \cdots, z]$. If

$$F=D\tilde{F}+\Delta F, \qquad \mathrm{mmc}(\Delta F)=0(\varepsilon),$$
$$G=D\tilde{G}+\Delta G, \qquad \mathrm{mmc}(\Delta G)=0(\varepsilon), \qquad (2)$$

then D is called an approximately common divisor, of accuracy ε, of F and G. In particular, if D is the largest-degree polynomial among the approximately common divisors of accuracy ε, it is called the approximate GCD, of accuracy ε, of F and G, and is denoted as

$$D=\gcd(F, G; \varepsilon). // \qquad (3)$$

Note 1. We usually normalize D. Below, if either F or G is not normalized, we denote the approximate GCD of F and G as $\gcd'(F, G; \varepsilon)$ instead of $\gcd(F, G; \varepsilon)$.

Note 2. If we change the value of ε, $\gcd(F, G; \varepsilon)$ may also change. Furthermore, for a given value of ε, the number of approximate GCDs is usually not one but infinite. For example, if $D=\gcd(F, G; \varepsilon)$ then $D+\Delta D$, with ΔD a polynomial such that $\mathrm{mmc}(\Delta D)=0(\varepsilon)$, is also an approximate GCD, of accuracy ε, of F and G. These complications do not cause any serious problems in actual application.

Difinition 5 [content and primitive part]. Let F be a normalized polynomial in $C[x, y, \cdots, z]$ and let ε be a small positive number. Among the approximate divisors of F that are included in $C[y, \cdots, z]$ and are of accuracy ε, the polynomial of largest degree *w.r.t.* $y, \cdots, z$ is called the content of F, of accuracy ε, and is denoted as $\mathrm{cont}(F; \varepsilon)$. With $\mathrm{cont}(F; \varepsilon)$, F can be decomposed as

$$F=\mathrm{cont}(F; \varepsilon)\tilde{F}+\Delta F, \qquad \mathrm{mmc}(\Delta F)=0(\varepsilon). \qquad (4)$$

$\tilde{F}$ is called the primitive part, of accuracy ε, of F and is denoted as $\mathrm{pp}(F; \varepsilon)$. //

Note 1. Like $\gcd(F, G; \varepsilon)$, $\mathrm{cont}(F; \varepsilon)$ and $\mathrm{pp}(F; \varepsilon)$ are not unique unless $\mathrm{cont}(F; \varepsilon)=1$.

Note 2. Let F be defined in (1); then $\mathrm{cont}(F; \varepsilon)$ can be calculated as

$$\gcd'(f_m, \gcd'(f_{m-1}, \cdots, \gcd'(f_1, f_0; \varepsilon); \cdots; \varepsilon); \varepsilon) \qquad (5)$$

where $\gcd'$ is a GCD operation for unnormalized polynomials. The GCD of unnormalized polynomials and its accuracy will be explained in Section 3.

3. Euclidean Algorithm for Approximate GCD

The classical method of calculating the GCD of polynomials F and G, with $F, G\in C[x, y, \cdots, z]$ and $\deg(F)\geqq\deg(G)$, is the Euclidean algorithm. This algorithm calculates the so-called polynomial remainder sequence (PRS) ($P_1=F$, $P_2=G$, $\cdots$, $P_k\neq 0$, $P_{k+1}=0$) by the iteration formula

$$\beta_i P_{i+1}=\mathrm{remainder}(\alpha_i P_{i-1}, P_i),\ i=2, 3, \cdots,$$
$$\alpha_i, \beta_i\in C[y, \cdots, z]. \qquad (6)$$

According to the subresultant theory of PRS [1, 2], P_i, $i\geqq 3$, can be represented by F, G, and their coefficients as

$$P_i=\lambda_i\begin{vmatrix} f_m\, f_{m-1} & \cdots\cdots & f_{2j-n+2} & x^{n-j-1}F \\ \cdot & & \cdot\quad\cdot & \cdot \\ & \cdot & \cdot\quad\cdot & \cdot \\ & f_m\, f_{m-1}\cdots & f_{j+1} & x^0F \\ g_n\, g_{n-1} & \cdots\cdots & g_{2j-m+2} & x^{m-j-1}G \\ \cdot & & \cdot\quad\cdot & \cdot \\ & \cdot & \cdot\quad\cdot & \cdot \\ & g_n\, g_{n-1}\cdots & g_{j+1} & x^0G \end{vmatrix}, \tag{7}$$

where $\lambda_i \in C(y, \cdots, z)$, $\deg(P_{i-1})=j+1$, and $f_k=g_k=0$ if $k<0$. The determinant in Eq. (7) is called the j-th order subresultant of F and G, and $|\lambda_i|=1$ if we choose

$$\alpha_i=\mathrm{lc}(P_i)^{d_i+1},\ d_i=\deg(P_{i-1})-\deg(P_i), \tag{8}$$

$$\beta_2=1,\ \gamma_2=1,$$

$$\beta_i=\mathrm{lc}(P_{i-1})\gamma_i^{d_{i-1}},\ \gamma_i=\mathrm{lc}(P_{i-1})^{d_{i-1}}\gamma_{i-1}{}^{1-d_{i-1}},\ i\geqq 3. \tag{9}$$

This choice of α_i and β_i is called the subresultant-PRS algorithm [1, 2].

The following theorem is crucial in calculating the approximate GCD:

Theorem 1. Let P_1 and P_2 be normalized polynomials in $C[x, y, \cdots, z]$, and let ε be a small positive number, $0<\varepsilon \ll 1$. Let $(P_1, P_2, P_3, \cdots)$ be the subresultant PRS and $\deg(P_k)=\deg(\gcd(P_1, P_2; \varepsilon))$; then

$$\mathrm{mmc}(P_{k+1})\leqq 0(\varepsilon). \tag{10}$$

(Proof) Putting $D=\gcd(P_1, P_2; \varepsilon)$, we have

$$P_i=D\tilde{P}_i+\varepsilon P_i',\ \mathrm{mmc}(P_i')\leqq 0(1),\ i=1, 2.$$

Assuming that ε is a parameter, we consider P_1 and P_2 to be polynomials in ε also. By assumption, we have

$$P_{k+1}(\varepsilon\rightarrow 0)=0.$$

Since the subresultant representing P_{k+1} is a determinant whose elements are linear in ε, and P_1 and P_2 are normalized, the above equation gives Eq. (10).//

Note that the reverse of Theorem 1 is not true, as Example 1 shows below.

In actual computations, the magnitude of the coefficients of the subresultant PRS may vary widely even if the initial polynomials are normalized and have no approximately common factor. In particular, such a phenomenon occurs in abnormal PRS, which is defined as follows:

Definition 6 [abnormal PRS]. Let $(P_1, P_2, P_3, \cdots)$ be a PRS generated by (6). The PRS is called abnormal if $\mathrm{mmc}(\mathrm{lc}(P_i))\ll \mathrm{mmc}(P_i)$ for some i.//

Example 1. Abnormal subresultant PRS.

$$P_1=(y^2+1)x^3+(y^2+1.001)x^2+(2y+1),$$

$$P_2=(y^2-1)x^3+(y^2-1.001)x^2+(y-3),$$

$$P_3=(y^2-1)P_1-(y^2+1)P_2$$
$$=(0.002y^2)x^2+(y^3+4y^2-3y+2),$$

$$P_4=[(0.002y^2)^2P_2-Q_3P_3]/(y^2-1)$$
$$=-(0.002y^2)(y^3+4y^2-3y+2)x$$
$$-(0.002y^2)(y^3+4y^2-3.003y+2.002),$$

where $Q_3=(0.002y^2)(y^2-1)x$
$+(0.002y^2)(y^2-1.001)$,

$$P_5=[(0.002y^2)^2(y^3+4y^2-3y+2)^2P_3-Q_4P_4]$$
$$/(0.002y^2)^2$$
$$=y^9+12.002y^8+39.016y^7-1.98001y^6+69.04y^5$$
$$+168.05006y^4-159.02404y^3+102.00801y^2-36y+8,$$

where $Q_4=-(0.002y^2)(y^3+4y^2-3y+2)x$
$+(0.002y^2)(y^3+4y^2-3.003y+2.002)/$

We see $\mathrm{mmc}(\mathrm{lc}(P_3))\ll \mathrm{mmc}(P_3)$, which makes $\mathrm{mmc}\ (P_4) =0(10^{-3})$ although $\gcd\ (P_1, P_2;\ 10^{-3})=1$.//

Theorem 1 suggests that we may decide $P_k \propto \gcd(P_1, P_2; \varepsilon)$ if $\mathrm{mmc}(P_{k+1})=o(\varepsilon)$. Example 1 shows, however that this criterion is wrong if $(P_1, P_2, P_3, \cdots)$ is chosen to be the subresultant PRS, and we must calculate remainders (and quotients) carefully. In the case of univariate polynomials, [5] presents a clever normalization of remainders so that $\mathrm{mmc}(P_{i+1})$ decreases markedly compared with $\mathrm{mmc}(P_i)$ only when P_i is an approximate divisor of P_1 and P_2. We generalize the normalization to the multivariate case.

Rule 1 [normalization of remainders in PRS]. Let F and G be polynomials in $C[x, y, \cdots, z]$, with $\deg(F) \geqq \deg(G)$. Let $\tilde{R}$ and $\tilde{Q}$ be polynomials such that

$$\tilde{R}=\mathrm{lc}(G)^{d+1}F-\tilde{Q}G,\quad \deg(\tilde{R})<\deg(G),$$

where $d=\deg(F)-\deg(G)$.

($\tilde{R}$ and $\tilde{Q}$ are called the pseudo-remainder and pseudo-quotient, respectively, satisfying $\tilde{R}, \tilde{Q} \in C[x, y, \cdots, z]$.) In the division of $\mathrm{lc}(G)^{d+1}F$ by G, we calculate the remainder R by normalizing $\tilde{R}$ *as*

$$R=\tilde{R}/\max\{\mathrm{mmc}[\mathrm{lc}(G)^{d+1}],\ \mathrm{mmc}(\tilde{Q})\}. \tag{11}$$

The remainder R is called the normalized pseudo-remainder of F and G, and is denoted as

$$R=\mathrm{prem}'\ (F/, G).// \tag{12}$$

By Rule 1, Theorem 1 is elaborated as follows:

Theorem 2. Let P_1, P_2, and ε be defined as in Theorem 1. Let PRS $(P, P_2, P_3, \cdots)$ be calculated as

$$\beta_iP_{i+1}=\mathrm{prem}'(P_{i-1}, P_i),\ i=2, 3, \cdots,$$

where $d_i=\deg(P_{i-1})-\deg(P_i)$ and

$$\beta_2=\gamma_2=1,$$

$$\beta_i=\mathrm{Normalize}[\mathrm{lc}(P_{i-1})\gamma_i^{d_{i-1}}],\ i\geqq 3,$$

$$\gamma_i=\mathrm{lc}(P_{i-1})^{d_{i-1}}\gamma_{i-1}^{1-d_{i-1}},\ i\geqq 3.$$

Then, $\mathrm{mmc}(P_{k+1})/\mathrm{mmc}(P_k)$ decreases markedly only when P_k is an approximate divisor of $\mathrm{lc}(P_k)^{d_k+1}P_{k-1}$. In particular, if $\deg(P_k)=\deg(\gcd(P_1, P_2; \varepsilon))$ then $\mathrm{mmc}(P_{k+1})\leqq 0(\varepsilon)$.

(Proof) F, G, and R in Rule 1 satisfy

$$R=[\mathrm{lc}(G)^{d+1}/\rho]F-[\tilde{Q}/\rho]G,$$

where $\rho=\max\{\mathrm{mmc}(\mathrm{lc}(G)^{d+1}, \mathrm{mmc}(\tilde{Q})\}$. Since $\max\{\mathrm{mmc}(\mathrm{lc}(G)^{d+1}/\rho), \mathrm{mmc}(\tilde{Q}/\rho)\}=1$, we see that $\mathrm{mmc}(R)\simeq\max\{\mathrm{mmc}(F), \mathrm{mmc}(G)\}$ unless cancellation occurs. We have, however, $\mathrm{mmc}(R)\ll\max\{\mathrm{mmc}(F), \mathrm{mmc}(G)\}$ iff the main parts of $\mathrm{lc}(G)^{d+1}F$ and $\tilde{Q}\,G$ cancel each other out. Note that P_1 and P_2 are polynomials whose coefficients are linear in ε and that $P_k(\varepsilon\to 0) \propto \gcd(P_1(\varepsilon\to 0), P_2(\varepsilon\to 0))$. Furthermore, the above formulas for PRS show that $\mathrm{mmc}(P_i)\leqq O(1)$, $i=1, 2, \cdots$. Hence, we see that $\mathrm{mmc}(P_{k+1})=O(\varepsilon)$. //

Let us see how well Rule 1 works for PRS in Example 1.

Example 1′. Normalized PRS of Example 1.

$$P_3=(0.002y^2)x^2+(y^3+4y^2-3y+2)$$
$$\tilde{P}_4=-(0.002y^2)(y^3+4y^2-3y+2)x$$
$$\quad -(0.002y^2)(y^3+4y^2-3.003y+2.002),$$
$$P_4=\tilde{P}_4/\max\{(0.002)^2, 0.002\times 1.001\}$$
$$=-(0.999y^2)(y^3+4y^2-3y+2)x$$
$$\quad -(0.999y^2)(y^3+4y^2-3.003y+2.002).$$

We see that, although $\mathrm{mmc}(P_4)=O(10^{-3})$ in Example 1, $\mathrm{mmc}(P_4)$ in Example 1′ is $O(4)$ on account of the normalization of remainders by Rule 1. //

Let us give another example in which $\gcd(P_1, P_2\colon \varepsilon)\neq$ constant.

Example 2. Normalized PRS with approx-GCD$=x+1$.

$$P_1=(x+1)[(y-1)x+1]+0.001(x+y)$$
$$=(y-1)x^2+(y+0.001)x+(1+0.001y),$$
$$P_2=(x+1)[(y+1)x-1]+0.001(yx-1)$$
$$=(y+1)x^2+(1.001y)x-1.001,$$
$$P_3=(y+1)P_1-(y-1)P_2$$
$$=(-0.001y^2+2.002y+0.001)x$$
$$\quad +(0.001y^2+2.002y-0.001),$$
$$\tilde{P}_4=(-0.001y^2+2.002y+0.001)^2P_2-\tilde{Q}_3P_3$$
$$=0.000002y^4+0.004002y^3-0.000002y^2-0.00801y-,$$
$$P_4=\tilde{P}_4/\max\{(2.002)^2, 1\}$$
$$=0.0009985y^3-0.0019985y+(\text{small terms}). \quad //$$

Example 2 reveals another problem in calculating the approximate GCD, that is, the appearance of many small terms in PRS; in Example 2, $P_3\sim\gcd(P_1, P_2; \varepsilon)\simeq x+1$, while P_3 contains four small terms. If we include these small terms in the GCD, the approximate GCD will become an ugly expression. Fortunately, the approximate GCD is not unique, but is ambiguous up to terms of magnitude $O(\varepsilon)$. We therefore discard such small terms in order to get approximate GCD's in simple forms, by imposing the following rule:

Rule 2 [rounding of P_k]. Let P_k be an element of PRS such that $P_k=\text{const}\times\gcd(P_1, P_2; \varepsilon)$. Then, after normalizing P_k as $\mathrm{mmc}(P_k)=1$, we round off the coefficients of P_k at $O(\varepsilon)$. (In the following, rounding is made at 2ε.) This rounding procedure is denoted as Roundoff $(P_k, 2\varepsilon)$. //

Example 2′. Application of Rule 2 to P_3 in Example 2. Normalization of P_3 gives

$$P_3'=P_3/2.002$$
$$=(-0.00049y^2+y+0.00049)x$$
$$\quad +(0.00049y^2+y-0.00049).$$

We see that $P_4(x)=O(\varepsilon(x))$ with $\varepsilon\simeq 10^{-3}$, and therefore we round off the coefficients of P_3' at $2\varepsilon\simeq 2\times 10^{-3}$, obtaining $P_3''=\mathrm{Roundoff}(P_3', 2\times 10^{-3})=yx+y$. This gives

$$\gcd(P_1, P_2\colon 10^{-3})=\mathrm{pp}(P_3''; 10^{-3})=x+1. \quad //$$

Now, we state a Euclidean algorithm for calculating the approximate GCD. Note that, in the actual computation, the value of ε is not known in advance but is usually determined by the PRS calculation. Considering this, as well as Rules 1 and 2, we modify the conventional Euclidean algorithm (the subresultant PRS algorithm) as follows:

```
Algorithm Approx-GCD($P_1$, $P_2$, $\varepsilon_0$)
Input: normalized polynomials $P_1$ and $P_2$ in $C[x, y, \cdots, z]$, with $\deg(P_1)\geqq\deg(P_2)$,
       a small positive number $\varepsilon_0$, $0<\varepsilon_0\ll 1$;
Output: $D=\gcd(P_1, P_2; \varepsilon)$ and $\varepsilon$, where $\deg(D)$ is the
        largest possible number satisfying $0<\varepsilon\leqq\varepsilon_0$;
        let $d_k=\deg(P_k)-\deg(P_{k-1})$, below;
Step 1: $k\leftarrow 2$; $\gamma\leftarrow 1$; $P_3\leftarrow\mathrm{prem}'(P_1, P_2)$;
Step 2: if $\mathrm{mmc}(P_{k+1})<\varepsilon_0$ then goto Step 3;
        if $\deg(P_{k+1})=0$ then return $(1, \varepsilon_0)$;
        $k\leftarrow k+1$; $\gamma\leftarrow\mathrm{lc}(P_{k-1})^{d_{k-1}}\gamma^{1-d_{k-1}}$;
        $\beta\leftarrow\mathrm{Normalize}(\mathrm{lc}(P_{k-1})\gamma^{d_{k-1}})$;
        $P_{k+1}\leftarrow\mathrm{prem}'(P_{k-1}, P_k)/\beta$; goto Step 2;
Step 3: $\varepsilon\leftarrow\mathrm{mmc}(P_{k+1})$; $P_k\leftarrow\mathrm{Normalize}(P_k)$;
        $P_k\leftarrow\mathrm{Roundoff}(P_k, 2\varepsilon)$;
Step 4: $D'\leftarrow\mathrm{pp}(P_k; \varepsilon)$;
        $C_1\leftarrow\mathrm{cont}(P_1; \varepsilon)$; $C_2\leftarrow\mathrm{cont}(P_2; \varepsilon)$;
        return $(D'\times\gcd(C_1, C_2; \varepsilon), \varepsilon)$ //
```

Note. In the calculation of pp and cont in Step 4, we cannot apply Approx-GCD recursively, because some coefficients of P_1 and P_2 may be unnormalized.

Let us now consider the calculation of $\mathrm{cont}(P; \varepsilon)$ and $\mathrm{pp}(P; \varepsilon)$, where we assume that P is normalized, i.e. $\mathrm{mmc}(P)=1$, and the value of ε is given. Let $\mathrm{cont}(P; \varepsilon)=c$, $c\in C[y, \cdots, z]$, $\mathrm{mmc}(c)=1$. Then c must be calculated so that it satisfies

$$P(x, y, \cdots, z)=c(y, \cdots, z)\tilde{P}(x, y, \cdots, z)$$
$$\quad +\Delta P(x, y, \cdots, z), \tag{13}$$
$$\mathrm{mmc}(\Delta P)=O(\varepsilon).$$

This means that, for every pair (f, g), where f and g are two different coefficients of P, we must calculate c so that

$$f=c\tilde{f}+\Delta f, \quad \mathrm{mmc}(\Delta f)\leqq O(\varepsilon),$$

$$g=c\tilde{g}+\Delta g, \quad \mathrm{mmc}(\Delta g)\leqq O(\varepsilon). \tag{14}$$

Using the algorithm Approx-GCD, we can calculate c to satisfy Eq. (14) as follows. Let $\mathrm{mmc}(f)=\alpha$ and $\mathrm{mmc}(g)=\beta$ and assume that $\alpha\geqq\beta$. Since P is normalized, we may further assume that $1\geqq\alpha\geqq\beta$. If $\beta\leqq\varepsilon$ then we can choose $c=f$, so we consider only the case in which $\beta>\varepsilon$. Putting $f'=f/\alpha$ and $g'=g/\beta$, we first calculate c' to satisfy

$$\mathrm{mmc}(c')=1,$$
$$f'=c'\tilde{f}'+\Delta f', \quad \mathrm{mmc}(\Delta f')\leqq O(\varepsilon/\beta),$$
$$g'=c'\tilde{g}'+\Delta g', \quad \mathrm{mmc}(\Delta g')\leqq O(\varepsilon/\beta).$$

The vaku of c' satisfying these equations is calculated by Approx-GCD(f', g', ε/β). We see that the second equation in (14) is satisfied by $c=c'$ or c=(any approximate divisor of c'). Therefore, we can calculate c as

$$(c, \varepsilon')=\text{Approx-GCD}(f/\alpha, c', \varepsilon/\alpha).$$

The above discussion is summarized in the following algorithm, which may be used to calculate cont.

```
Algorithm Approx-GCD'($f$, $g$, $\varepsilon$)
Input: $f, g\in[y, \cdots, z]$ such that $\mathrm{mmc}(f)\leqq 1$, $\mathrm{mmc}(g)\leqq 1$,
        a small positive number $\varepsilon$, $0<\varepsilon\ll 1$;
Output: $c=\gcd(f, g; \varepsilon')$, where $\deg(c)$ is the largest
        possible number satisfying $0<\varepsilon'\leqq\varepsilon$;
  if $\mathrm{mmc}(g)>\mathrm{mmc}(f)$ then return Approx-GCD'($g, f, \varepsilon$);
  if $\mathrm{mmc}(g)\leqq\varepsilon$ then return $f$;
  $(c', \varepsilon')\leftarrow$Approx-GCD($f/\mathrm{mmc}(f)$, $g/\mathrm{mmc}(g)$, $\varepsilon/\mathrm{mmc}(g)$);
  if $c'=1$ then return 1;
  $(c, \varepsilon')\leftarrow$Approx-GCD($f/\mathrm{mmc}(f)$, $c'$, $\varepsilon/\mathrm{mmc}(g)$)
  return $c$. //
```

Example 3. Approximate GCD of unnormalized polynomials:

$f=(y+z)(y+z+0.01)+0.001(y-z)$, $\mathrm{mmc}(f)=2.0$,
$g=0.1(y+z)^2+0.001(y-1)$, $\mathrm{mmc}(g)=0.2$,
$\varepsilon=2\times 10^{-3}$.
$(c', \varepsilon')=\text{Approx-GCD}(f/2.0, g/0.2, \varepsilon/0.2)$
$=(y^2/2+yz+z^2/2, 0.01)$.
$(c, \varepsilon')=\text{Approx-GCD}(f/2.0, c', \varepsilon/2.0)$
$=(y+z, 0.01)$. //

Using cont(P; ε), we can calculate pp(P; ε) easily. However, it should be noted that P usually contains small terms, as the above examples show. Hence, $Q=P/\mathrm{cont}(P; \varepsilon)$ will also contain small terms. We want to discard such small terms so as to obtain pp(P; ε) in a simple form. We therefore impose the following rule:

Rule 3. Given normalized polynomials F and G such that $F=QG+\Delta F$, $\mathrm{mmc}(\Delta F)=O(\varepsilon)$, we calculate Q as $Q=\mathrm{Roundoff}(F/G, 2\varepsilon)$. //

Thus, given F and G in $C[x, y, \cdots, z]$, we decompose F and G as in (2) in the following way:

```
Algorithm Decompose($F$, $G$, $\varepsilon_0$)
Input: normalized polynomials $F$ and $G$ in $C[x, y, \cdots, z]$,
        a small positive number $\varepsilon_0$, $0<\varepsilon_0\ll 1$;
Output: $(\tilde{F}, \Delta F, \tilde{G}, \Delta G)$ such that
        $F=D\tilde{F}+\Delta F$,  $\mathrm{mmc}(\Delta F)\leqq O(\varepsilon_0)$,
        $G=D\tilde{G}+\Delta G$,  $\mathrm{mmc}(\Delta G)\leqq O(\varepsilon_0)$;
  $(D, \varepsilon)\leftarrow$Approx-GCD($F$, $G$, $\varepsilon_0$);
  if $D=1$ then return ($F$, 0, $G$, 0);
  $\tilde{F}\leftarrow$Roundoff($F/D$, $2\varepsilon$); $\Delta F\leftarrow F-D\tilde{F}$;
  $\tilde{G}\leftarrow$Roundoff($G/D$, $2\varepsilon$); $\Delta G\leftarrow G-D\tilde{G}$;
  return ($\tilde{F}$, $\Delta F$, $\tilde{G}$, $\Delta G$). //
```

4. A kind of Ill-Conditioned System of Equations

Let us now apply the approximate GCD calculation to solving an ill-conditioned system of algebraic equations, such as the one given in section 1. Since the system may be given in an arbitrary form, we first consider how to regularize it.

Definition 7 [regular set of polynomials]. Let F_i, $i=1, \cdots, r$, be polynomials in $C[x, y, \cdots, z]$ and let $F_i=\Sigma_j f_{ij}(y, \cdots, z)x^j$. The set $\{F_1, \cdots, F_r\}$ is called regular if

$$\mathrm{mmc}(\mathrm{lc}(F_i))=O(1),\ i=1, \cdots, r, \tag{15}$$
$$\max\{\mathrm{mmc}(f_{ij})\,|\,i=1, \cdots, r, j=0, \cdots, \deg(F_i)\}=O(1) \text{ or } 0. \ //$$

Any set of polynomials $\{F_1, \cdots, F_r\}$ can be transformed into a regular set $\{F_1', \cdots, F_r'\}$ by the transformation

$$F_i'(x, y, \cdots, z)\leftarrow\eta_i F(\xi x, y, \cdots, z),\ i=1, \cdots, r, \tag{16}$$

where η_i, $i=1, \cdots, r$, and ξ are suitably chosen numbers. We determine the values of η_i and ξ as follows:

```
Algorithm Regularize($\{F_1, \cdots, F_r\}$)
Input: a set of polynomials $\{F_1, \cdots, F_r\}\in C[x, y, \cdots, z]$;
Output: regular set $\{F_1', \cdots, F_r'\}$ by the transformation (16);
  for $i=1, \cdots, r$, determine $\xi_i$ so that
    if $\mathrm{lc}(F_i)=F_i$ then $\xi_i\leftarrow 0$
      else $\mathrm{mmc}[\mathrm{lc}(F_i(\xi_i x, \cdots))]=\mathrm{mmc}[F_i(\xi_i x, \cdots)-\mathrm{lt}(F_i(\xi_i x, \cdots))]$;
  $\xi\leftarrow\max(\xi_1, \cdots, \xi_r)$; if $\xi=0$ then $\xi\leftarrow 1$;
  for $i=1, \cdots, r$, determine $\eta_i$ so that
    $\mathrm{mmc}[\eta_i\,\mathrm{lc}(F_i(\xi x, \cdots))]=1$;
  return $\{\eta_1 F_1(\xi x, \cdots), \cdots, \eta_r F_r(\xi x, \cdots)\}$. //
```

Example 4. Regularization of $\{F_1, F_2\}$.

$F_1=(y^2+1)x^2+(5y-3)x+(100y^2+50y+3)$,
$F_2=(2y-1)x^2+(y+2)x+(y^2+y+1)$.

The ξ_1 and ξ_2 in the algorithm Regularize are determined as

$F_1(\xi_1 x, y)$: $\xi_1^2=\max\{5\xi_1, 100\}\rightarrow\xi_1=10$,
$F_2(\xi_2 x, y)$: $2\xi_2^2=\max\{2\xi_2, 1\}\rightarrow\xi_2=1$.

Hence, $\xi=\max\{\xi_1, \xi_2\}=10$ and we find $\eta_1=0.01$, $\eta_2=0.005$. Thus, we obtain

$F_1' = (y^2+1)x^2 + (0.5y-0.3)x + (y^2+0.5y+0.03)$,
$F_2' = (y-0.5)x^2 + (0.05y+0.01)x + 0.005(y^2+y+1)$.
//

Before discussing the general case, we consider a system of two equation $\{F=0,\ G=0\}$, where we assume that $\{F,\ G\}$ is regular and satisfies

$$F = D\tilde{F} + \Delta F, \quad \mathrm{mmc}(\Delta F) = 0(\varepsilon),$$
$$G = D\tilde{G} + \Delta G, \quad \mathrm{mmc}(\Delta G) = 0(\varepsilon). \tag{17}$$

Suppose that we have already decomposed F and G as above, and that we wish to consider the system $\{F=0,\ H=0\}$, where

$$H = F\tilde{G} - G\tilde{F} = \tilde{G}\Delta F - \tilde{F}\Delta G. \tag{18}$$

Since $H = F\tilde{G} - G\tilde{F} = -G\tilde{F}$ when $F=0$, we can decompose the system $\{F=0,\ H=0\}$ as

$$\{F=0,\ H=0\} = \{F=0,\ G=0\} \vee \{F=0,\ \tilde{F}=0\}.$$

Hence, the system $\{F=0,\ H=0\}$ contains all the roots of the original system $\{F=0,\ G=0\}$ and the roots of an extra system $\{F=0,\ \tilde{F}=0\}$. Although the system $\{F=0,\ G=0\}$ is ill-conditioned for numerical methods, as we have mentioned in Section 1, we may reasonably expect the system $[F=0,\ H=0\}$ to be no more ill-conditioned than $[F=0,\ G=0\}$, because $\tilde{F}$ and $\tilde{G}$ have no approximately common factor.

The above transformation of an ill-conditioned system gives us a useful method of determining good approximations of the roots; in solving a system of algebraic equations by Newton's iteration method, giving good initial values of the roots is quite important but not easy in practice, particularly for ill-conditioned systems.

We assume that the system $\{F=0,\ H=0\}$ is no more ill-conditioned than $\{F=0,\ G=0\}$. Then, we may determine the approximations of the roots of the system $\{F=0,\ G=0\}$ by neglecting ΔF in F, that is, by setting $F \simeq D\tilde{F}$. This gives us

$$\{D\tilde{F}=0,\ H=0\} = \{D=0,\ H=0\} \vee \{\tilde{F}=0,\ H=0\}.$$

Since $H = F\tilde{G} - G\tilde{F} = F\tilde{G}$ when $\tilde{F}=0$, we have

$$\{\tilde{F}=0,\ H=0\} = \{\tilde{F}=0,\ F=0\} \vee \{\tilde{F}=0,\ \tilde{G}=0\}.$$

The system $\{\tilde{F}=0,\ F=0\}$ is irrelevant to the system $\{F=0,\ G=0\}$ as we have noted above, and therefore we discard it. Thus, we determine the approximations of the roots by solving

$$\{D=0,\ H=0\} \text{ and } \{\tilde{F}=0,\ \tilde{G}=0\}. \tag{19}$$

The solutions of the first system in (19) correspond to the roots of the original system that are very difficult to obtain by conventional numerical methods, while the solutions of the second system correspond to the roots that can be obtained by conventional methods without much difficulty.

It should be noted that the above method does not always give the approximations of all the roots. For example, consider the system $\{F_1=0,\ F_2=0\}$, where

$$F_1(x, y) = x(y-1)^2 - 0.01(y-1),$$
$$F_2(x, y) = xy(y-1) + 0.01y(y+0.001x).$$

We see that $(x=-1000,\ y-1)$ is a root of this system. Using the algorithm Decompose, we find

$$F_1 = x(y-1)\tilde{F}_1 + \Delta F_1,$$
$$F_2 = x(y-1)\tilde{F}_2 + \Delta F_2,$$

where $\tilde{F}_1,\ \Delta F_1,\ \tilde{F}_2,\ \Delta F_2$ are given by

$$\tilde{F}_1 = y-1, \quad \Delta F_1 = -0.01(y-1),$$
$$\tilde{F}_2 = y, \quad \Delta F_2 = 0.01y(y+0.001x).$$

According to the above scheme, the approximations of the roots are determined by $\{\tilde{F}_1=0,\ \tilde{F}_2=0\}$ and $\{D=0,\ H=0\}$, where

$$D = x(y-1),$$
$$H = y(y-1)(y+0.001x+1).$$

The system $\{\tilde{F}_1=0,\ \tilde{F}_2=0\}$ has no root, and $[D=0,\ H=0\}$ has an infinite number of roots on the plane $y=1$. This means that the system $\{D=0,\ H=0\}$ does not give good approximations of the root $(x=-1000,\ y=1)$. The method mentioned above will, however, give good approximations for roots whose magnitudes are less than $0(1)$, because we may reasonably ignore ΔF in F in this case.

We now consider a system of r equations $\{F_1=0, \cdots, F_r=0\}$, where we assume that $\{F_1, \cdots, F_r\}$ is regular and that the system is ill-conditioned in the sense mentioned in Section 1, namely that for some i and j, $i \neq j$, F_i and F_j have an approximate GCD D_{ij}, $D_{ij} \neq$ constant. Generalizing the method mentioned above, we transform $\{F_1, \cdots, F_r\}$ in the following way. Using F_1, we first transform $F_2, \cdots, F_r$ to $F_2', \cdots, F_r'$, successively, as mentioned above. Then, using F_2', we transform $F_3', \cdots, F_r'$, and so on.

```
Algorithm Trans-Eqs($\{F_1, \cdots, F_r\}, \varepsilon_0$)
Input: regular set $\{F_1, \cdots, F_r\} \in C[x, y, \cdots, z]$,
        a small positive number $\varepsilon_0$, $0 < \varepsilon_0 \ll 1$;
Output: set of polynomials $\{F_1', \cdots, F_r'\}$, where
        $\{F_1'=0, \cdots, F_r'=0\}$ will be mostly well-con-
        ditioned ($F_i'$ and $F_j'$, $1 \leq i \neq j \leq r$, do not have an
        approximately common factor of accuracy $\varepsilon_0$);
    for $i=1, \cdots, r-1$ do begin
        for $j=i+1, \cdots, r$ do begin
            ($\tilde{F}, \Delta F, \tilde{G}, \Delta G$)$\rightarrow$Decompose($F_i, F_j, \varepsilon_0$);
            If $\Delta F \neq 0$ then $F_j \leftarrow$Normalize ($\tilde{G}\Delta F - \tilde{F}\Delta G$)}
        end;
    end;
    return $\{F_1, \cdots, F_r\}$. //
```

The system $\{F_1'=0, \cdots, F_r'=0\}$ calculated by Trans-Eqs may still be ill-conditioned in the sense mentioned in Section 1, but we may well expect that it is mostly well-conditioned. If, however, the transformed system is not well-conditioned, we can apply Trans-Eqs further.

5. Examples of Ill-Conditioned Algebraic Systems

Let us see how well the method described in Section 4 works. We show two examples, one in $C[x, y]$ and the other in $C[x, y, z]$. For each system, we calculate the roots by the following three methods:

Method I. A mostly algebraic method, described in Ref. 3. We derive a univariate polynomial $G(x)$ by calculating the Gröbner basis of the ideal $(F_1, \cdots, F_r)$. Then, we solve $G(x)=0$ by subdividing the two-dimensional space of a complex variable x. The roots of $G(x)=0$ are substituted into the other elements of the Gröbner basis to determine the roots of $\{F_1=0, \cdots, F_r=0\}$. Since this method can calculate the roots quite accurately (we calculate them to 15 significant digits below), we use this method to check the correctness of numerical methods.

Method II. A numeric method using the conventional form of Newton's iteration. Determination of the initial values for the iterative calculation will be explained in each example below.

Method III. A numeric-algebraic method using our technique for transforming an ill-conditioned system to a well-conditioned one. The initial values of the roots for the iterative calculation are determined as explained in Section 4.

Example 5.

$$F_1=(x^2+y^2-1)(xy-0.25)+0.0001xy,$$
$$F_2=(x^2+y^2-1)(x-y)-0.00001(x+1).$$

Using the algorithm Decompose given in Section 3, we can decompose F_1 and F_2 as

$$F_1=D\tilde{F}_1+\Delta F_1,\ F_2=D\tilde{F}_2+\Delta F_2,$$
$$D=x^2+y^2-1,$$
$$\tilde{F}_1=xy-0.25,\quad \Delta F_1=0.0001xy,$$
$$\tilde{F}_2=x-y,\quad \Delta F_2=-0.00001(x+1).$$

Hence,

$$\begin{aligned}F_2'&=\text{Normalize}(\tilde{F}_2\Delta F_1-\tilde{F}_1\Delta F_2)\\&=\text{Normalize}[0.0001(1.1x^2y-xy^2+0.1xy\\&\quad-0.025x-0.025)]\\&=(1.1x^2y-xy^2+0.1xy-0.025x-0.025)/1.1.\end{aligned}$$

The initial values for Method II are determined by solving the following systems of equations:

$$\{\tilde{F}_1=0,\ \tilde{F}_2=0\},\ \{D=0,\ \tilde{F}_1=0\},\ \{D=0,\ \tilde{F}_2=0\}.$$

Note that the first system gives the same initial values as those for Method III, and that the roots of the second and the third systems make $F_1\to\Delta F_1$ and $F_2\to\Delta F_2$. Hence, the initial values are quite good in the practical sense.

Example 6.

$$F_1=(x^2+y^2+z^2-1)(x-y-z)-0.000001(x+z+1),$$
$$F_2=(x^2+y^2+z^2-1)(x+y+z+0.5)-0.000001(x+1),$$
$$F_3=(x^2+y^2+z^2-1)(xyz-0.25)-0.000001xyz.$$

We can decompose F_1, F_2, and F_3 as

$$F_1=D\tilde{F}_1+\Delta F_1,\ F_2=D\tilde{F}_2+\Delta F_2,\ F_3=D\tilde{F}_3+\Delta F_3,$$
$$D=x^2+y^2+z^2-1,$$
$$\tilde{F}_1=x-y-z,\quad \Delta F_1=-0.000001(x+z+1),$$
$$\tilde{F}_2=x+y+z+0.5,\quad \Delta F_2=-0.000001(x+1),$$
$$\tilde{F}_3=xyz-0.25,\quad \Delta F_3=0.000001xyz.$$

Hence,

$$\begin{aligned}F_2'&=\text{Normalize}(\tilde{F}_2\Delta F_1-\tilde{F}_1\Delta F_2)\\&=(-(2y+3z+0.5)x-(y+z)(z+2)\\&\quad-0.5z-0.5)/2,\\F_3'&=\text{Normalize}(\tilde{F}_3\Delta F_1-\tilde{F}_1\Delta F_3)\\&=(2x^2yz+(-y^2z+yz-0.25)x-0.25z-0.25)/2.\end{aligned}$$

Instead of the original system, we may solve

$$\{F_1=0,\ F_2'=0,\ F_3'=0\},\tag{20}$$

which contains all the roots of the original system. Putting $\Delta F_1=0$, we approximate this system as

$$\begin{aligned}\{F_1=,\ F_2'=0,\ F_3'=0\}&\simeq\{\tilde{F}_1D=0,\ F_2'=0,\ F_3'=0\}\\&=\{\tilde{F}_1=0,\ F_2'=0,\ F_3'=0\}\vee\{D=0,\ F_2'=0,\ F_3'=0\}\end{aligned}$$

Using the relation $F_j'=\tilde{F}_jF_1-\tilde{F}_1F_j$, $j=2$, 3, we have

$$\{\tilde{F}_1=0,\ F_2'=0,\ F_3'=0\}=\{\tilde{F}_1=0,\ \tilde{F}_2F_1=0,\ \tilde{F}_3F_1=0\}.$$

The r.h.s. system can be split into four subsystems, three of which are irrelevant to the original system. Hence, we determine the initial values of the roots by solving

$$\{\tilde{F}_1=0,\ \tilde{F}_2=0,\ \tilde{F}_3=0\}\vee\{D=0,\ F_2'=0,\ F_3'=0\}.\tag{21}$$

We use two types of initial values for Method II. The first type consists of the same initial values as those for Method III, which are rough solutions of the systems in (21). The second type consists of random numbers distributed in the real domain $-1\leqq x, y, z\leqq 1$.

The results of root calculation are shown in Tables 1 and 2, where no result is given for the calculation of Example 6 by Method I. We found that calculation of the Gröbner basis of Example 6 took too many hours, so we gave up attempting to solve Example 6 by Method I. As this experience shows, algebraic methods often take too long to be used in solving practical problems. Let us explain Tables 1 and 2 separately.

Table 1 shows that the problem is very ill-conditioned for the conventional numeric method (Method II), although some roots are not very hard to calculate (roots corresponding to the solutions of $\{\tilde{F}_1=0, \tilde{F}_2=0\}$). In fact, Method II missed some roots (those of Nos. 5 and 6), and the accuracy of the roots around $D=0$ is not good in spite of so many iterations. On the other hand, Method III calculated every root quite easily with

Table 1 Results of root calculation by three methods. Method I is algebraic (to check the correctness of the roots calculated by Methods II and III); Method II is numeric, using the conventional form of Newton's method, and Method III is algebraic-numeric, using our technique for transforming an ill-conditioned system to well-conditioned one. The numbers in parentheses in the right-hand columns for Methods II and III show the numbers of iterations before convergence. The calculations were done by using Maple on a SUN-4.

No.	root	Method I	Method II		Method III	
1	x =	0.9990732	0.9990724		0.9990732	
	y =	0.0432840	0.0433031	(19)	0.0432840	(2)
2	x =	-0.0262280	-0.0262224		-0.0262280	
	y =	0.9996512	0.9996513	(17)	0.9996512	(2)
3	x =	-1.0000000	-1.0000000		-1.0000000	
	y =	0.0	-3.3942D-5	(17)	3.428D-22	(1)
4	x =	-0.0227383	-0.0227343		-0.0227384	
	y =	-0.9997464	-0.9997465	(18)	-0.9997464	(2)
5	x =	0.6645091			0.6645091	
	y =	0.7471453			0.7471453	(3)
6	x =	-0.7141434			-0.7141434	
	y =	-0.6998564			-0.6998564	(3)
7	x =	0.5000350	0.5000350		0.5000350	
	y =	0.5000650	0.5000650	(3)	0.5000650	(2)
8	x =	-0.5000550	-0.5000550		-0.5000550	
	y =	-0.5000450	-0.5000450	(2)	-0.5000450	(2)
Elapsed time		549.9 sec	0.352 sec		122.89 sec	

Table 2 Results of real root calculation by Methods II and III (Method I did not terminate within a reasonable time). Two types of initial values are used for Method II: values of the first kind are determined by Eq. (21) and are the same as those for Method III, and the results are shown in the left three columns for Method II; values of the second kind are chosen randomly, and the results are shown in the right two columns for Method II. We have chosen 100 sets of random numbers as initial values of (x, y, z) and performed Newton's iteration. Among them, 31 cases converged to the No. 1 root with an average of 24.90 iterations, 36 cases converged to the No. 2 root, with an average of 29.53 iterations, and so on. However, 32 cases did not converge within 100 iterations. The numbers in parentheses show the (average) numbers of iterations before convergence.

No.	root	Method II				Method III	
		initial=Eq.(21)		initial = random		final	
		initial value	iter	trials 100	iter (average)	value	iter
1	x =	-0.2500				-0.2499989	
	y =	-1.1327	(3)	31	(24.90)	-1.1327841	(2)
	z =	0.8827				0.8827837	
2	x =	-0.2500				-0.2499998	
	y =	0.8827	(3)	36	(29.53)	0.8827823	(2)
	z =	-1.1327				-1.1327818	
3	x =	-1.0000				-1.0000000	
	y =	0.0000	(1)	1	(76.00)	3.3951D-9	(1)
	z =	0.0000				2.779D-17	
4	x =	-0.8639				-0.8639101	
	y =	-0.4941	(15)	0	—	-0.4941624	(2)
	z =	-0.0972				-9.7276D-2	
5	x =	-0.5239				-0.5239150	
	y =	-0.4336	(4)	0	—	-0.4336388	(2)
	z =	-0.7331				-0.7331235	

Table 3 Final check of the roots of Eq. (20): Substitute each root into the original system $\{F_1=0, F_2=0, F_3=0\}$ (and $\tilde{F}_1$, $\tilde{F}_2$ and $\tilde{F}_3$). Roots ex-1 and ex-2 are not the roots of the original system.

Root No.	Residuals after substitution of 'roots' to					
	F_1	F_2	F_3	$\tilde{F}_1$	$\tilde{F}_2$	$\tilde{F}_3$
1	0.59D-15	0.52D-15	-0.62D-15	0.14D-5	0.66D-6	-0.22D-6
2	0.93D-15	-0.46D-16	-0.63D-15	0.34D-6	0.66D-6	-0.22D-6
3	-0.26D-16	-0.13D-16	-0.66D-17	-1.0	-0.5	-0.25
4	0.31D-15	0.11D-14	0.34D-15	-0.2724	-0.9553	-0.2195
5	-0.13D-14	0.25D-14	0.89D-15	0.6428	-0.1190	0.6428
ex-1	-0.37D-21	-0.74D-6	0.17D-15	0.16D-14	-1.077	-0.3462
ex-2	-0.15D-20	0.88D-7	0.57D-15	0.2 D-14	-1.077	-0.3462

good accuracy. We show the computation elapsed times in Table 1. Compared with Method II, Method III took a considerable time, most of which was spent in calculating an approximate GCD. As we will note in the next section, our current GCD algorithm is quite inefficient, and we have a good chance of reducing the computation elapsed time of Method III by employing a better GCD algorithm.

We show only the results of real root calculation in Table 2. The table shows the usefulness of our method even more clearly. Solving (21) roughly, we obtain 7 real solutions. Among them, five solutions are shown in Table 2. Other two are

$$(x, y, z) = (-0.2113,\ 0.5773,\ -0.7886)$$

and

$$(-0.7886,\ -0.5773,\ -0.2113).$$

Solving (20) by Newton's method with these roots as initial approximations, we obtain the following solutions:

$$\text{ex}-1\text{: } (-0.2113247,\ 0.5773506,\ -0.7886753)$$

and

$$\text{ex}-2\text{: } (-0.7886750,\ -0.5773501,\ -0.2113249).$$

As mentioned above, we must check each root of (21) whether or not it is also a root of the original system $\{F_1=0,\ F_2=0,\ F_3=0\}$. The results of substitutions of the roots obtained into the original system are shown in Table 3. It is apparent that the roots No. 1 to No. 5

satisfy the original system but the roots ex-1 and ex-2 do not. As we can see from the results for Method II, it is very hard to obtain roots No. 3 to No. 5 by conventional Newton's iteration with random number initial values. Thus, with conventional Newton's method, finding all the roots is extremely difficult if the system is ill-conditioned. On the other hand, with our Method III, even such ill-conditioned systems as Examples 5 and 6 are easily solved. This means also that our method gives very good initial values for the iteration.

6. Final Remarks

The system of algebraic equations considered in this paper is of a special kind and does not appear very frequently in practical problems. However, we may claim to have clearly demonstrated the usefulness of the approximate GCD operation for multivariate polynomials, and that of approximate algebrain operations in general.

In order to handle more general kinds of ill-conditioned systems of algebraic equations than that considered in this paper, we are now investigating approximate algebraic computation more widely in other algebraic operations.

For the computation of the multivariate polynomial GCD, the Euclidean algorithm that we employed in this paper is much less efficient than the EZ GCD algorithm [4] and the PC-PRS GCD algorithm [7]. If we can use such efficient GCD algorithms to calculate the approximate GCD efficiently, it will become possible to handle large systems of algebraic equations.

Reference

1. Brown, W. S. and Traub, J. F. On Euclid's Algorithm and the Theory of Subresultants, *J. ACM*, **18** (1971), 505-514.
2. Collins, G. E. Subresultants and Reduced Polynomial Remainder Sequences, *J. ACM*, **14** (1967), 128-142.
3. Davenport, J. H., Siret, Y. and Tounier, E. *Computer Algebra*, Academic Press (1988), 95-122.
4. Moses, J. and Yun, D. Y. Y. The EZ GCD Algorithm, *Proc. ACM Nat. Conf.* (1973), 159-166.
5. Sasaki, T. and Noda, M. T. Approximate Square-free Decomposition and Root-finding of Ill-conditioned Algebraic Equations, *J. Int. Proces.* **12** (1989), 159-168.
6. Sasaki, T. and Sasaki, M. Analysis of Accuracy Decreasing in Polynomial Remainder Sequence with Floating-point Number Coefficients, *J. Inf. Proces.* **12** (1989), 394-403.
7. Sasaki, T. and Suzuki, M. Three New Algorithms for Multivariate Polynomial GCD, J. Symb. Comp., **12** (1991), in press.
8. Schönhage, A. Quasi-GCD Computations, *J. Complexity*, **1** (1985), 118-137.
9. Tanabe, K. Centered Newton Method for Solving a System of Nonlinear Equations: Global Method, preprint of Inst. Statistical Math. (1988).

(Received November 8, 1989)