The Subresultant PRS Algorithm

W. S. BROWN
Bell Laboratories

Two earlier papers described the generalization of Euclid's algorithm to deal with the problem of computing the greatest common divisor (GCD) or the resultant of a pair of polynomials. A sequel to those two papers is presented here

In attempting such a generalization one easily arrives at the concept of a polynomial remainder sequence (PRS) and then quickly discovers the phenomenon of explosive coefficient growth. Fortunately, this explosive growth is not inherent in the problem, but is only an artifact of various naive solutions If one removes the content (that is, the GCD of the coefficients) from each polynomial in a PRS, the coefficient growth in the resulting primitive PRS is relatively modest. However, the cost of computing the content (by applying Euclid's algorithm in the coefficient domain) may be unacceptably or even prohibitively high, especially if the coefficients are themselves polynomials in one or more additional variables

The key to controlling coefficient growth without the costly computation of GCD's of coefficients is the fundamental theorem of subresultants, which shows that every polynomial in a PRS is proportional to some subresultant of the first two. By arranging for the constants of proportionality to be unity, one obtains the subresultant PRS algorithm, in which the coefficient growth is essentially linear. A corollary of the fundamental theorem is given here, which leads to a simple derivation and deeper understanding of the subresultant PRS algorithm and converts a conjecture mentioned in the earlier papers into an elementary remark.

A possible alternative method of constructing a subresultant PRS is to evaluate all the subresultants directly from Sylvester's determinant via expansion by minors A complexity analysis is given in conclusion, along lines pioneered by Gentleman and Johnson, showing that the subresultant PRS algorithm is superior to the determinant method whenever the given polynomials are sufficiently large and dense, but is inferior in the sparse extreme

Key Words and Phrases symbolic algebra, subresultant, polynomial remainder sequences, Euclid's algorithm, greatest common divisor, greatest common factor, resultant, Sylvester determinant, coefficient growth, analysis of algorithms
CR Categories 5 7, 5 25

1. INTRODUCTION

This paper is a sequel to two earlier papers [1, 2] on the generalization of Euclid's algorithm to deal with the problem of computing the greatest common divisor (GCD) or the resultant of a pair of polynomials. In attempting such a generalization one easily arrives at the concept of a polynomial remainder sequence

Author's address· Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974

(PRS), and then quickly discovers the phenomenon of explosive coefficient growth.

Fortunately, this explosive growth is not inherent in the problem, but is only an artifact of various naive solutions. If one removes the content (that is, the GCD of the coefficients) from each polynomial in a PRS, the coefficient growth in the resulting primitive PRS is relatively modest. However, the cost of computing the content (by applying Euclid's algorithm in the coefficient domain) may be unacceptably or even prohibitively high, especially if the coefficients are themselves polynomials in one or more additional variables.

The key to controlling coefficient growth without the costly computation GCD's of coefficients is the discovery by Collins [4] that every polynomial in a PRS is proportional to some subresultant of the first two. By arranging for the constants of proportionality to be unity, Collins developed the subresultant PRS algorithm, which is the subject of this paper. Unfortunately, Collins' formulation of the algorithm was too complicated for convenient application, and he therefore recommended the simpler reduced PRS algorithm as a practical compromise.

Later, Brown and Traub [2] discovered the fundamental theorem of subresultants, and used it to obtain a much simpler formulation of the subresultant PRS algorithm. Also, Brown [1] derived essentially linear bounds on the coefficient growth in a subresultant PRS (using a then unpublished theorem of Goldstein and Graham [6]), while showing that the coefficient growth in a reduced PRS can be exponential if the sequence involves degree differences greater than unity. Although such abnormal sequences are a set of measure zero in the space of all PRS's, they are not uncommon in practice, and it is important to deal sensibly with them when they arise.

A few months after [1] and [2] were published, I discovered a corollary of the fundamental theorem, which led to a simpler derivation and deeper understanding of the subresultant PRS algorithm. The new approach, which is presented in this paper, reveals the subresultant PRS algorithm as a simple generalization of the reduced PRS algorithm and converts the conjecture that was mentioned in [1] and [2] into an elementary remark.

Although I cannot assert with confidence that the subresultant PRS algorithm is optimal for any important class of GCD problems, it is clearly the best of its kind and deserves to be thoroughly understood. Among its competitors are the modular GCD algorithm, discussed in [1], and the EZ-GCD algorithm of Moses and Yun [10], which is also modular. Both of these algorithms have the overwhelming advantage that the GCD, which is almost certain to be smaller than the given polynomials, can be computed without ever forming the associated subresultant, which is likely to be very much larger. However, for small problems both suffer from complexity, while for sparse problems the modular GCD algorithm suffers from fill-in. Furthermore, one can construct problems on which the EZ-GCD algorithm performs poorly, and such problems might conceivably be important in practice.

If one's objective is to compute the resultant of a pair of polynomials rather than their GCD, or if the degrees of the given polynomials are not too large, then it may be advantageous to evaluate all of the subresultants directly from Sylvester's determinant via expansion by minors. The merits of this approach are

explored empirically by Ku and Adler [9], and their important but overstated conclusions are challenged by Collins [3].

Section 2 of this paper establishes notation and restates the fundamental theorem of subresultants without proof. Section 3 presents the new formulation of the subresultant PRS algorithm, and Section 4 illustrates it with a familiar example. Section 5 reviews the improved PRS algorithm of [1] and shows how to introduce two specific improvements that are clearly worthwhile. Finally, Section 6 analyzes the computing time of the algorithm and compares it with the determinant method mentioned above.

2. THE FUNDAMENTAL THEOREM

Let $\mathbf{I}$ be a unique factorization domain in which there is some way of finding GCD's, and let $\mathbf{I}[x]$ denote the domain of polynomials in x with coefficients in $\mathbf{I}$. Assuming that the terms of a polynomial $F \in \mathbf{I}[x]$ are arranged in the order of decreasing exponents, the first term is called the leading term; its coefficient $\mathrm{lc}(F)$ is called the leading coefficient, and its exponent $\partial(F)$ is called the degree.

Since the familiar process of polynomial division with remainder requires exact divisibility in the coefficient domain, it is usually impossible to carry it out for nonzero $A, B \in \mathbf{I}[x]$. However, the process of pseudodivision [8, p. 369] always yields a unique pseudoquotient $Q = \mathrm{pquo}(A, B)$ and pseudoremainder $R = \mathrm{prem}(A, B)$, such that $b^{\delta+1}A - QB = R$ and $\partial(R) < \partial(B)$, where b is the leading coefficient of B and $\delta = \partial(A) - \partial(B)$.

For nonzero $A, B \in \mathbf{I}[x]$, we say that A is similar to B ($A \sim B$) if there exist nonzero $\alpha, \beta \in \mathbf{I}$ such that $\alpha A = \beta B$. Here α and β are called coefficients of similarity.

For nonzero $F_1, F_2 \in \mathbf{I}[x]$ with $\partial(F_1) \geq \partial(F_2)$, let $F_1, F_2, \ldots, F_{k+1}$ be a sequence of polynomials such that $F_i \sim \mathrm{prem}(F_{i-2}, F_{i-1})$ for $i = 3, \ldots, k+1$, and $F_{k+1} = \mathrm{prem}(F_{k-1}, F_k) = 0$. Such a sequence is called a polynomial remainder sequence (PRS). From the definitions, it follows that there exist nonzero $\alpha_i, \beta_i \in \mathbf{I}$ and $Q_i \sim \mathrm{pquo}(F_{i-2}, F_{i-1})$ such that

$$\beta_i F_i = \alpha_i F_{i-2} - Q_i F_{i-1}, \qquad \partial(F_i) < \partial(F_{i-1}), \quad i = 3, \ldots, k+1. \tag{1}$$

Because of the uniqueness of pseudodivision, the PRS beginning with F_1 and F_2 is unique up to similarity. Furthermore, it is easy to see that $\gcd(F_1, F_2) \sim \gcd(F_2, F_3) \sim \cdots \sim \gcd(F_{k-1}, F_k) \sim F_k$. Thus the construction of the PRS yields the desired GCD to within similarity.

Let $n_i = \partial(F_i)$ for $i = 1, \ldots, k$, and note that $n_1 \geq n_2 > \cdots > n_k \geq 0$. Let $\delta_i = n_i - n_{i+1}$ for $i = 1, \ldots, k-1$, and note that $\delta_1 \geq 0$, while $\delta_i > 0$ for $i > 1$. If $\delta_i = 1$ for all $i > 1$, the PRS is called normal; otherwise it is called abnormal. Finally, let $f_i = \mathrm{lc}(F_i)$ for $i = 1, \ldots, k$, and let

$$\alpha_i = f_{i-1}^{\delta_{i-2}+1}, \quad i = 3, \ldots, k+1, \tag{2}$$

so that eq. (1) becomes

$$\beta_i F_i = \mathrm{prem}(F_{i-2}, F_{i-1}), \quad i = 3, \ldots, k+1. \tag{3}$$

When a method for choosing the β_i is given, this equation and the terminating condition $F_{k+1} = 0$ constitute an algorithm for constructing the PRS.

Next, let $R(j)$ denote the jth subresultant of F_1 and F_2 for $0 \leq j < n_2$, as defined in [2] and [4]. It is easy to show that $R(j)$ is a polynomial of degree at most j, each of whose coefficients is a determinant of order $n_1 + n_2 - 2j$ with coefficients of F_1 and F_2 as its elements, and in particular that $R(0)$ is the classical resultant, $\text{res}(F_1, F_2)$. With our notation and definitions now established, we are at last prepared to state the fundamental theorem, which shows in detail how the PRS elements F_i and the nonzero subresultants $R(j)$ are similar.

THEOREM 1. *Let $F_1, F_2, \ldots, F_{k+1}$ be a PRS in $\mathbf{F}[x]$ with F_1, F_2 in $\mathbf{I}[x]$, where $\mathbf{F}$ is the quotient field of $\mathbf{I}$. Then for $i = 3, \ldots, k$,*

$$R(n_{i-1} - 1) = \gamma_i F_i \equiv G_i, \tag{4}$$

$$R(j) = 0, \quad n_{i-1} - 1 > j > n_i, \tag{5}$$

$$R(n_i) = \theta_i F_i \equiv H_i, \tag{6}$$

$$R(j) = 0, \quad n_k > j \geq 0, \tag{7}$$

where

$$\gamma_i = (-1)^{\sigma_i} f_{i-1}^{1-\delta_{i-1}} \prod_{l=3}^{i} (\beta_l/\alpha_l)^{n_{l-1}-n_{i-1}+1} f_{l-1}^{\delta_{l-2}+\delta_{l-1}}, \tag{8}$$

$$\theta_i = (-1)^{\tau_i} f_i^{\delta_{i-1}-1} \prod_{l=3}^{i} (\beta_l/\alpha_l)^{n_{l-1}-n_i} f_{l-1}^{\delta_{l-2}+\delta_{l-1}}, \tag{9}$$

with

$$\sigma_i = \sum_{l=3}^{i} (n_{l-2} - n_{i-1} + 1)(n_{l-1} - n_{i-1} + 1), \tag{10}$$

$$\tau_i = \sum_{l=3}^{i} (n_{l-2} - n_i)(n_{l-1} - n_i). \tag{11}$$

Remarks. This theorem accounts for all of the $R(j)$, $n_2 > j \geq 0$. It should be noted that eq. (5) is vacuous when $\delta_{i-1} \leq 2$, and furthermore that eqs. (4) and (6) are identical when $\delta_{i-1} = 1$. Finally, we extend eqs. (6), (9), and (11) to $i = 2$ by defining

$$R(n_2) = f_2^{\delta_1 - 1} F_2 \equiv H_2 \tag{12}$$

which is also suggested by the definition of $R(j)$ as a determinant.

3. THE SUBRESULTANT PRS ALGORITHM

In this section we present, verify, and discuss a recursive formula for computing β_i such that $F_i = G_i$ for $i = 3, \ldots, k$. We then recast it in the form of an algorithm, which is called the subresultant PRS algorithm.

Referring to eq. (4), our goal is to choose the β_i so that $\gamma_3 = \quad = \gamma_k = 1$. For reasons that will become clear, it is helpful to define β_{k+1} and γ_{k+1} by extending eq. (8) to $i = k + 1$ and requiring that $\gamma_{k+1} = 1$, even though β_{k+1} has no significance in eq. (3) and γ_{k+1} does not even appear in eq. (4). Note that the total exponent of f_k in γ_{k+1} is $\delta_{k-1} + 1$; thus γ_{k+1} does not depend on the undefined

quantity δ_k, whose appearance is only superficial.

THEOREM 2. *To obtain a PRS with* $\gamma_i = 1$ *for* $i = 3, \ldots, k+1$, *it suffices to choose*

$$\beta_3 = (-1)^{\delta_1+1} \tag{13}$$

$$\beta_i = (-1)^{\delta_{i-2}+1} f_{i-2} h_{i-2}^{\delta_{i-2}}, \quad i = 4, \ldots, k+1, \tag{14}$$

where

$$h_2 = f_2^{\delta_1} \tag{15}$$

$$h_i = f_i^{\delta_{i-1}} h_{i-1}^{1-\delta_{i-1}}, \quad i = 3, \ldots, k. \tag{16}$$

PROOF. We begin by setting $\gamma_3 = 1$ to obtain eq. (13). Then for $i = 4, \ldots, k+1$, we set $\gamma_i/\gamma_{i-1} = 1$ to obtain eq. (14) with

$$h_i = (-1)^{n_1-n_i+i-1} f_i^{-1} \prod_{l=3}^{i+1} (\alpha_l/\beta_l), \tag{17}$$

for $i = 2, \ldots, k-1$, and we define h_k by extending this to $i = k$. Setting $i = 2$ in eq. (17) yields eq. (15), and finally dividing h_i by h_{i-1} for $i = 3, \ldots, k$ yields

$$h_i/h_{i-1} = (-1)^{\delta_{i-1}+1} (f_{i-1}/f_i)(\alpha_{i+1}/\beta_{i+1}) \tag{18}$$

which in turn yields eq. (16).

THEOREM 3. *The auxiliary quantities* $h_2, \ldots, h_k$ *defined in Theorem* 2 *satisfy the relation*

$$h_i = \theta_i f_i = lc(H_i), \quad i = 2, \ldots, k. \tag{19}$$

PROOF. First, note that

$$\theta_i f_i = \theta_i f_i/\gamma_{i+1}, \quad i = 2, \ldots, k, \tag{20}$$

since eqs. (13) through (16) were chosen to make all the $\gamma_i = 1$. Replacing θ_i by eq. (9) and γ_{i+1} by eq. (8), we find that the right side of eq. (20) is equal to the right side of eq. (17), and therefore that $h_i = \theta_i f_i$. The identity $\mathrm{lc}(H_i) = \theta_i f_i$ follows immediately from eq. (6).

Remark. From eq. (19) and the fact that H_i is a subresultant, it follows immediately that all of the h_i, and hence also all of the β_i, are in **I**, as conjectured in [1] and [2].

Algorithm 1 (*Subresultant PRS*). Given primitive polynomials F_1, F_2 in $\mathbf{I}[x]$, we can obtain the subresultant PRS, $G_1 = F_1$, $G_2 = F_2$, $G_3, \ldots, G_k$, by computing

$$G_3 = (-1)^{\delta_1+1} \mathrm{prem}(G_1, G_2),$$

$$G_i = \frac{(-1)^{\delta_{i-2}+1} \mathrm{prem}(G_{i-2}, G_{i-1})}{g_{i-2} h_{i-2}^{\delta_{i-2}}}, \quad i = 4, \ldots, k, \tag{21}$$

where $g_i = \mathrm{lc}(G_i)$ for $i = 1, \ldots, k$, and

$$h_2 = g_2^{\delta_1}, \qquad h_i = g_i^{\delta_{i-1}} h_{i-1}^{1-\delta_{i-1}}, \quad i = 3, \ldots, k. \tag{22}$$

The iteration stops because $\text{prem}(G_{k-1},G_k) = 0$. Then if $n_k > 0$, we have

$$\gcd(F_1,F_2) = \text{pp}(G_k), \qquad \text{res}(F_1,F_2) = 0, \tag{23}$$

where pp denotes the primitive part, while if $n_k = 0$, we have

$$\gcd(F_1,F_2) = 1, \qquad \text{res}(F_1,F_2) = h_k. \tag{24}$$

PROOF. First we replace β_i in eq. (3) by eqs. (13) and (14). Then, using Theorem 2, we replace F_i by G_i and f_i by g_i for all $i \geq 1$.

Remarks. This algorithm should be compared with Collins' reduced PRS algorithm, which is defined by

$$F_3 = \text{prem}(F_1,F_2), \qquad F_i = \frac{\text{prem}(F_{i-2},F_{i-1})}{f_{i-2}^{\delta_{i-3}+1}}, \quad i = 4, \ldots, k+1. \tag{25}$$

For a normal sequence, Algorithm 1 clearly specifies the same computation except possibly for signs. In the general case, the denominator of eq. (21) is a simple generalization of the denominator of eq. (25). Note that an abnormality at step $i+1$ (that is, $n_{i+1} < n_i - 1$, or in other words $\delta_i > 1$) yields a denominator of higher degree at step $i+2$ in the subresultant PRS algorithm (21), but does not do so until step $i+3$ in the reduced PRS algorithm (25).

To understand the significance of eq. (22), we divide through by h_{i-1}, and thus obtain

$$h_i/h_{i-1} = (g_i/h_{i-1})^{\delta_{i-1}}, \quad i = 3, \ldots, k, \tag{26}$$

which shows that $h_i = \text{lc}(R(n_i))$ is obtained from $h_{i-1} = \text{lc}(R(n_{i-1}))$ and $g_i = \text{lc}(R(n_{i-1} - 1))$ by geometric extrapolation. To visualize this, suppose **I** is the domain of integers (or any other domain of real numbers), and plot $\text{lc}(R(j))$ versus j on semilog paper. Then the points corresponding to h_{i-1}, g_i, and h_i for any fixed i will lie on a straight line.

The algorithm as presented does not explicitly compute the subresultants $H_3, \ldots, H_k$. However, since $H_i \sim G_i$ by the fundamental theorem, it suffices to compute

$$H_i = h_i G_i/g_i, \quad i = 3, \ldots, k \tag{27}$$

at the end of step i.

4. AN EXAMPLE

Let us consider again the example

$$\begin{aligned} F_1(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5 \\ F_2(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21 \end{aligned} \tag{28}$$

from [3, pp. 370–371] and [1]. For convenience we first compute the primitive PRS, $P_1 = F_1$, $P_2 = F_2$, $P_3, \ldots, P_k$, defined by

$$P_i = R_i/\beta_i \tag{29}$$

where

$$R_i = \text{prem}(P_{i-2},P_{i-1}), \qquad \beta_i = \text{content}(R_i) \tag{30}$$

for $i = 3, \ldots, k$. The results, showing only the coefficients for brevity, are

i	n_i	β_i	P_i
1	8		1, 0, 1, 0, −3, −3, 8, 2, −5
2	6		3, 0, 5, 0, −4, −9, 21
3	4	−3	5, 0, −1, 0, 3
4	2	−45	13, 25, −49
5	1	−50	4663, −6150
6	0	$7 \times 13^3 \times 9311$	1

(31)

With this table and the identity

$$\text{prem}(aA,bB) = ab^{\delta+1}\,\text{prem}(A,B) \tag{32}$$

where $\delta = \partial(A) - \partial(B)$, it is easy to compute the PRS that is determined from F_1 and F_2 by any proposed algorithm. In particular, Algorithm 1 yields

i	G_i/P_i	H_i/P_i
1	1	—
2	1	3
3	3	5
4	5	13
5	2	2
6	$2^2 \times 7 \times 9311$	$2^2 \times 7 \times 9311$

(33)

where the P_i are given in (31).

5. THE IMPROVED PRS ALGORITHM

We shall now review the improved PRS algorithm proposed in [1] and show how to realize two particular improvements whenever the opportunity arises.

In carrying out the subresultant PRS algorithm it may happen that a divisor γ_i of the content of G_i is available with little or no extra work. If so, we would like to compute the improved PRS, $F_1, F_2, \ldots, F_k$, such that

$$G_i = \gamma_i F_i, \quad i = 3, \ldots, k. \tag{34}$$

Making this substitution in eqs. (21) and (22) and using (32), we obtain a second algorithm.

Algorithm 2 (*Improved PRS*).

$$\begin{aligned} \gamma_3 F_3 &= (-1)^{\delta_1+1}\,\text{prem}(F_1,F_2) \\ \gamma_i F_i &= \frac{(-\gamma_{i-1})^{\delta_{i-2}+1}\,\text{prem}(F_{i-2},F_{i-1})}{f_{i-2}h_{i-2}^{\delta_{i-2}}}, \quad i = 4, \ldots, k+1, \end{aligned} \tag{35}$$

where

$$h_2 = f_2^{\delta_1}; \qquad h_i = \gamma_i^{\delta_{i-1}} f_i^{\delta_{i-1}} h_{i-1}^{1-\delta_{i-1}}, \quad i = 3, \ldots, k. \tag{36}$$

The simplest way to realize an improvement is to define

$$\gamma = \gcd(f_1, f_2) \tag{37}$$

and note that $\gamma | R(j)$ for all $j < n_2$, since each of these subresultants is a determinant with one row whose only nonzero elements are f_1 and f_2. In particular, $\gamma | G_i$ and $\gamma | H_i$ for $i = 3, \ldots, k$. Replacing γ_i by γ in eqs. (35) and (36), and introducing

$$\bar{h}_i = h_i/\gamma, \quad i = 2, \ldots, k, \tag{38}$$

we obtain a third algorithm.

Algorithm 3.

$$\begin{aligned} F_3 &= (-1)^{\delta_1+1} \operatorname{prem}(F_1, F_2)/\gamma \\ F_i &= \frac{(-1)^{\delta_{i-2}+1} \operatorname{prem}(F_{i-2}, F_{i-1})}{f_{i-2}(\bar{h}_{i-2})^{\delta_{i-2}}}, \quad i = 4, \ldots, k, \end{aligned} \tag{39}$$

where

$$\bar{h}_2 = f_2^{\delta_1} \gamma^{-1}; \qquad \bar{h}_i = f_i^{\delta_{i-1}} (\bar{h}_{i-1})^{1-\delta_{i-1}}, \quad i = 3, \ldots, k. \tag{40}$$

Note that Algorithm 3 is identical to Algorithm 1 except for the starting conditions; F_3 and $\bar{h}_2$ are smaller than G_3 and h_2, respectively, by a factor of γ. Also note that $\bar{h}_i \in \mathbf{I}$ for $i = 3, \ldots, k$, but $\bar{h}_2 \in \mathbf{I}$ only when $\delta_1 > 0$ or $\gamma = 1$.

To realize another improvement, we note with Hearn [7] that pseudodivision sometimes requires fewer than the expected number of subtraction steps and introduces some power of the leading coefficient of the divisor into the pseudoremainder as an extraneous factor. Furthermore, one can readily modify the pseudodivision process to compute this factor and the rest of the pseudoremainder as separate outputs. Accordingly, we define the modified pseudoremainder

$$\operatorname{mprem}(F_{i-2}, F_{i-1}) = f_{i-1}^{-\epsilon_{i-1}} \operatorname{prem}(F_{i-2}, F_{i-1}), \quad i = 3, \ldots, k, \tag{41}$$

where ϵ_{i-1} is the number of steps saved ($0 \leq \epsilon_{i-1} \leq \delta_{i-2}$).

Since γ, defined in eq. (37), can always be included in γ_i, we also define

$$\bar{\gamma}_i = \gamma_i/\gamma, \quad i = 3, \ldots, k. \tag{42}$$

Substituting eqs. (38), (41), and (42) into eqs. (35) and (36), we obtain a fourth algorithm.

Algorithm 4.

$$\begin{aligned} \bar{\gamma}_3 F_3 &= (-1)^{\delta_1+1} f_2^{\epsilon_2} \operatorname{mprem}(F_1, F_2)/\gamma \\ \bar{\gamma}_i F_i &= \frac{(-\bar{\gamma}_{i-1})^{\delta_{i-2}+1} f_{i-1}^{\epsilon_{i-1}} \operatorname{mprem}(F_{i-2}, F_{i-1})}{f_{i-2}(\bar{h}_{i-2})^{\delta_{i-2}}}, \quad i = 4, \ldots, k+1, \end{aligned} \tag{43}$$

where

$$\bar{h}_2 = f_2^{\delta_1} \gamma^{-1}; \qquad \bar{h}_i = (\bar{\gamma}_i)^{\delta_{i-1}} f_i^{\delta_{i-1}} (\bar{h}_{i-1})^{1-\delta_{i-1}}, \quad i = 3, \ldots, k. \tag{44}$$

It remains to choose the $\bar{\gamma}_i$. First, we reduce the fraction $f_2^{e_2}/\gamma$ to lowest terms and set $\bar{\gamma}_3$ equal to the resulting numerator. The modified pseudoremainder is then divisible by the denominator, and their quotient is $\pm F_3$. Similarly, for $i > 3$, we would like to reduce the fraction

$$(\bar{\gamma}_{i-1})^{\delta_{i-2}+1} f_{i-1}^{e_{i-1}}/f_{i-2}(\bar{h}_{i-2})^{\delta_{i-2}} \tag{45}$$

to lowest terms and set $\bar{\gamma}_i$ equal to the resulting numerator, but this may involve GCD computations too costly to be justified. As an alternative, we could expand the numerator of eq. (43), divide out the factors of the denominator, and then set $\bar{\gamma}_i$ equal to the largest product of powers of $\bar{\gamma}_{i-1}$ and f_{i-1} that divides the result. As another alternative, we could "grow" the numerator a factor at a time, dividing out denominator factors whenever possible, and set $\bar{\gamma}_i$ equal to the product of any unused powers of $\bar{\gamma}_{i-1}$ and f_{i-1}. This strategy is appealing because trial divisions are rarely costly unless they succeed. However, the optimal approach remains a topic for future research.

6. TIME COMPLEXITY

We now present a qualitative analysis of the time complexity of the subresultant PRS algorithm. Our purpose is not to obtain rigorous bounds on the computing time, but to acquire insight into the behavior of the algorithm. In particular, this analysis explains the surprising empirical observations of Ku and Adler [9], and resolves the apparent contradiction between those observations and the theoretical results of Collins [3]. Our technique is patterned after that used by Gentleman and Johnson [5] in their study of the evaluation of determinants, and our conclusions are remarkably similar.

In assessing costs, we shall assume that classical algorithms are used for multiplication, division, and pseudodivision. That is, we reject fast Fourier transforms, modular techniques, and so forth, in keeping with the fact that the subresultant PRS algorithm is itself classical. Thus the cost of computing $P = AB$ (measured in byte multiplications or word multiplications and ignoring other operations) is simply

$$C(AB) = \text{size}(A)\text{size}(B), \tag{46}$$

where the size function measures the total storage space (in bytes or words) that is required for its argument. Similarly, the cost of computing an exact quotient $Q = A/B$ is

$$C(A/B) = \text{size}(Q)\text{size}(B). \tag{47}$$

To compute $R = \text{prem}(A,B)$ in the general case, one constructs the sequence

$$R_0 = A, R_1, \ldots, R_\delta, R_{\delta+1} = R, \tag{48}$$

where

$$R_i = bR_{i-1} - r_{i-1}Bx^{\delta+1-i} \tag{49}$$

with $b = \text{lc}(B)$, $r_i = \text{lc}(R_i)$, and $\delta = \partial(A) - \partial(B)$. As we progress through the

sequence, the degrees of the R_i decrease, but the coefficients nearly always grow. Since the leading terms of bR_{i-1} and $r_{i-1}B$ need not be computed, the cost of each of these products is bounded by $\partial(A)\text{size}(b_{\max})\text{size}(r_{\max})$, where $b_{\max}$ and $r_{\max}$ are the largest coefficients of B and R, respectively. Hence

$$C(\text{prem}(A,B)) < 2(\delta + 1)\partial(A)\text{size}(b_{\max})\text{size}(r_{\max}). \tag{50}$$

Since the polynomials $G_3, \ldots, G_k$ are subresultants, their coefficients are determinants of coefficients of $G_1 = F_1$ and $G_2 = F_2$. Accordingly, we introduce $S(l,m)$ to denote the size of a product of m such determinants, each of order l. Letting l_i denote the order of the determinants that represent the coefficients of G_i, we have

$$l_1 = 1; \qquad l_2 = 1; \qquad l_i = n_1 + n_2 - 2(n_{i-1} - 1), \quad i = 3, \ldots, k, \tag{51}$$

which is an increasing function of i, and represents the loss of degree as we move through the PRS from G_1 and G_2 to G_i. From this definition it is easy to show that

$$1 \leqslant i - 2 \leqslant \tfrac{1}{2} l_i \leqslant n_1, \quad i = 3, \ldots, k, \tag{52}$$

and

$$\sum_{j=1}^{i-2} \delta_j \leqslant l_i - 2, \quad i = 3, \ldots, k. \tag{53}$$

To construct G_i via eq. (21), we must compute the pseudoremainder and then the quotient. By eq. (50) the cost of the first step is

$$C_i^{(1)} < 2(\delta_{i-2} + 1)n_{i-2}S(l_{i-1},1)S(l_i,\delta_{i-2} + 2), \tag{54}$$

while by eq. (47), the cost of the second step is

$$C_i^{(2)} < (n_i + 1)S(l_i,1)S(l_{i-1},\delta_{i-2} + 1). \tag{55}$$

Letting C_i denote the sum of these costs, it follows that

$$C_i < (2\delta_{i-2} + 3)n_{i-2}S(l_i,1)S(l_i,\delta_{i-2} + 2). \tag{56}$$

Models of Computation

To explore the significance of this result for the total cost

$$C = \sum_{i=3}^{k} C_i, \tag{57}$$

we shall consider the two extreme models of polynomial computation that are proposed in [5]. As stated there, these models "are extreme in the sense that other models tend to lie between them" and "we suggest that most practical problems will show aspects of both models."

In the dense model, G_1 and G_2 may be quite large, but the coefficients grow rather slowly as we proceed through the PRS. By contrast, in the sparse model,

G_1 and G_2 must be quite small compared with dense polynomials with comparable degree vectors, but the coefficient growth in the PRS is extremely rapid. Because of this rapid growth G_1 and G_2 must be reasonably small in an absolute sense too, or we will be unable to compute the PRS because of our limited resources. Of course, the sparseness in the sparse case will decrease as we proceed through the PRS, but we assume (for the sake of extremism) that G_1 and G_2 are so extremely sparse that the rapid coefficient growth implied by the sparse model is sustained to the very end.

We shall now specify the two models more precisely, and obtain a bound on C for each of them. In both models we assume, as in [5], that the addition or multiplication of polynomials never generates zero coefficients. Our only important departure from [5] is that we measure costs in byte multiplications or word multiplications rather than in integer multiplications, thus allowing for the growth of integer coefficients. As stated in [5], this change makes both models a little less extreme; by giving up a small measure of simplicity and "purity," we hope to have made our results more realistic and convincing.

Dense Model

In the dense model, we assume that the G_i are completely dense polynomials in one variable with integer coefficients. Thus the sum and product of polynomials with t_1 and t_2 terms, respectively, are both as small as possible; the sum has $\max(t_1,t_2)$ terms, while the product has $t_1 + t_2 - 1$. For convenience we approximate the bound on the size of a sum of integers by the maximum of their sizes; this approximation tends to underestimate the growth of integer coefficients and thereby makes our model slightly "denser" than reality. We also note that the size of a product of integers is the sum of their sizes. It now follows that

$$S(l,m) = lms, \tag{58}$$

where s is the size of a coefficient of G_1 or G_2. Substituting this into (56), we find

$$C_i < s^2 l_i^2 n_{i-2}(\delta_{i-2} + 2)(2\delta_{i-2} + 3) \leqslant s^2 l_i^3 n_{i-2}(2\delta_{i-2} + 3) \leqslant s^2 l_k^3 n_1(2\delta_{i-2} + 3), \tag{59}$$

and therefore

$$C < s^2 l_k^3 n_1[2(l_k - 2) + 3(k - 2)] < \tfrac{7}{2} s^2 l_k^4 n_1. \tag{60}$$

For a normal PRS with $\delta_1 \leqslant 1$, we can easily derive a tighter bound. In this case, eq. (56) yields

$$C_i < 15\, s^2 l_i^2 n_{i-2} \leqslant 15\, s^2 l_k^2 n_1, \tag{61}$$

and therefore

$$C < \tfrac{15}{2}\, s^2\, l_k^2 n_1(k - 2) \leqslant \tfrac{15}{2}\, s^2\, l_k^3 n_1, \tag{62}$$

in agreement with the result

$$C = O(s^2 n_1^4) \tag{63}$$

previously published in [1] and [3] for the normal univariate case.

Sparse Model

In the sparse model, we assume the G_i are polynomials in some main variable with extremely sparse, multivariate, polynomial coefficients, and we assume further that no combinations whatever occur when these coefficient polynomials are added or multiplied together. Thus the sum and product of polynomials with t_1 and t_2 terms, respectively, are both as large as possible; the sum has $t_1 + t_2$ terms, while the product has $t_1 t_2$. Since a determinant of order l is a sum of $l!$ products, each with l factors, it follows that

$$S(l,1) = lsl!\, t^l, \tag{64}$$

and more generally that

$$S(l,m) = lms(l!\, t^l)^m, \tag{65}$$

where the coefficients of G_1 and G_2 are t-term polynomials with integer coefficients of size s. Since the determinants of interest in this paper are in fact highly structured, the assumptions of the sparse model cannot be fully satisfied, and this formula is certainly too extreme. However, we are using the model primarily as a bound, and in that role it can certainly help us to acquire valuable insight.

Substituting eq. (65) into eq. (56), we find

$$\begin{aligned} C_i &< s^2 l_i^2 n_{i-2}(\delta_{i-2} + 2)(2\delta_{i-2} + 3)(l_i!\, t^{l_i})^{\delta_{i-2}+3} \\ &< 2s^2 l_i^4 n_{i-2}(l_i!\, t^{l_i})^{\delta_{i-2}+3} \\ &\leqslant 2s^2 l_k^4 n_1 (l_i!\, t^{l_i})^{\delta+3}, \end{aligned} \tag{66}$$

where

$$\delta = \max(\delta_1, \ldots, \delta_{k-2}) \leqslant l_k - 2. \tag{67}$$

In summing over (66), the last term dominates, and it is easy to show that

$$C < 2.0001\, s^2 l_k^4 n_1 (l_k!\, t^{l_k})^{\delta+3}, \tag{68}$$

which is our final upper bound. To obtain a lower bound, we note that

$$C \geqslant C_k > C_k^{(2)} = \text{size}(G_k)\text{size}(g_{k-2} h_{k-2}^{\delta_{k-2}}), \tag{69}$$

and therefore that

$$C > (n_k + 1)S(l_k,1)S(l_{k-2},\delta_{k-2} + 1) \geqslant S(l_k,1)S(l_{k-2}, 2) > 2s^2 l_{k-2}^2\, (l_{k-2}!\, t^{l_{k-2}})^3. \tag{70}$$

Although this lower bound is somewhat less fearsome than eq. (68), it remains severely superexponential.

7. CONCLUSION

The chief drawback of the subresultant PRS algorithm is that the last pseudo-remainder, $\beta_k G_k$, is substantially larger (overwhelmingly so in the sparse case) than the desired subresultant, G_k, which must be obtained from $\beta_k G_k$ and β_k by a very costly division. This phenomenon, which we call overshoot, could be avoided by expressing G_k as a determinant (see [2]) and evaluating it by the

method of nested minors (see [5]). Since each subresultant, $R(j)$, is a minor of the next larger one, $R(j-1)$, we could compute the entire PRS as a byproduct of computing the resultant $R(0)$.

In the dense extreme, this determinant method involves the evaluation of exponentially many minors, while the total cost of the subresultant PRS algorithm is only of order $s^2{n_1}^4$ (or $s^2{n_1}^5$ in abnormal cases), as discussed above.

However, in the sparse extreme, the situation is remarkably different. For the determinant method it follows from [5] that the unit cost (that is, the cost per unit size of G_k) is less than $3s$, while for the subresultant PRS algorithm it follows from eq. (69) that the unit cost exceeds

$$S(l_{k-2},2) = 2sl_{k-2}(l_{k-2}!\, t^{l_{k-2}})^2. \tag{71}$$

Since the assumptions of the sparse model cannot be fully satisfied in this context (see eq. (65) and the ensuing discussion), the actual difference between the two methods is certainly less extreme than this comparison suggests. A deeper understanding will require further study, involving less extreme models of sparseness and supported by detailed empirical evidence.

In their brief examination of dense polynomials in one variable, Ku and Adler [9] conclude that the subresultant PRS algorithm is better than the determinant method, but their examples are too small to reveal the substantial magnitude of the difference. In the rest of their study, they confine their attention to rather small polynomials in several variables (well removed from both of the extremes of this paper) and conclude that the determinant method (starting from Bezout's determinant rather than Sylvester's) is superior. I believe it is fair to say that this important practical observation and the controversy that followed deserve a large share of the credit for the theoretical results of [5] and the present paper.

REFERENCES

1 BROWN, W S On Euclid's algorithm and the computation of polynomial greatest common divisors *J ACM 18*, 4 (Oct. 1971), 478–504.

2 BROWN, W S, AND TRAUB, J.F. On Euclid's algorithm and the theory of subresultants. *J. ACM 18*, 4 (Oct. 1971), 505–514.

3 COLLINS, G E Comment on a paper by Ku and Alder. *Comm. ACM 12*, 6 (June 1969), 302–303.

4 COLLINS, G E Subresultants and reduced polynomial remainder sequences *J ACM 14*, 1 (Jan. 1967), 128–142.

5 GENTLEMAN, W M, AND JOHNSON, S C Analysis of algorithms A case study· Determinants of matrices with polynomial entries *ACM Trans Math Software 2*, 3 (Sept 1976), 232–241.

6. GOLDSTEIN, A J, AND GRAHAM, R.L A Hadamard-type bound on the coefficients of a determinant of polynomials *SIAM Rev 16* (July 1974), 394–395.

7. HEARN, A.C An improved non-modular GCD algorithm. SIGSAM Bull. (ACM) *6* (July 1972), 10–15

8. KNUTH, D E *The Art of Computer Programming, Vol 2.* Addison-Wesley, Reading, Mass 1969.

9 KU, S Y, AND ADLER, R J Computing polynomial resultants Bezout's determinant vs. Collins' reduced PRS algorithm. *Comm ACM 12*, 1 (Jan. 1969), 23–30.

10 MOSES, J, AND YUN, D Y Y. The EZ GCD algorithm Proc ACM Nat. Conf., Aug 1973, pp. 159–166.

Received December 1976, revised September 1977