The Subresultant Polynomial Remainder Sequence Algorithm

Ruiyuan (Ronnie) Chen

March 23, 2013

1 Introduction

Computation of polynomial greatest common divisors (GCDs) is an essential subproblem in many algorithms for computer algebra. For example, arithmetic with rational functions requires computing the GCD in order to keep the numerator and denominator coprime. In this article we introduce the problem of efficiently computing the GCD of two polynomials in $R[x]$ for some ring R.

If k is a field, then the GCD of two polynomials in $k[x]$ can be computed using the classical Euclidean algorithm. However, over a general ring R, the division algorithm, and thus the Euclidean algorithm, can fail. For example, over $\mathbb{Z}[x]$, with

$$f_1(x) = x^3 - 7x + 6, \qquad f_2(x) = 2x^2 - 5x + 3,$$

the division algorithm produces

$$f_1(x) = f_2(x)(\tfrac{1}{2}x + \tfrac{5}{4}) + (-\tfrac{9}{4}x + \tfrac{9}{4});$$

neither the quotient nor the remainder is in $\mathbb{Z}[x]$, although $\gcd(f_1, f_2) = x - 1$.

One obvious solution is to work over the fraction field of the coefficient ring, in this case $\mathbb{Q}$. However, the following example (which appears to be standard in the literature [1, 4, 6]) illustrates the phenomenon of *coefficient growth*.

Example 1. With

$$\begin{aligned} f_1(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5, \\ f_2(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21, \end{aligned}$$

the Euclidean algorithm in $\mathbb{Q}[x]$ produces the following sequence of remainders:

$$\begin{aligned} f_3(x) &= f_1(x) - (\tfrac{1}{3}x^2 - \tfrac{2}{9})f_2(x) = -\tfrac{5}{9}x^4 + \tfrac{1}{9}x^2 - \tfrac{1}{3}, \\ f_4(x) &= f_2(x) - (-\tfrac{27}{5}x^2 - \tfrac{252}{25})f_3(x) = -\tfrac{117}{25}x^2 - 9x + \tfrac{441}{25}, \\ f_5(x) &= f_3(x) - (\tfrac{125}{1053}x^2 - \tfrac{3125}{13689}x + \tfrac{51175}{59319}) = \tfrac{233150}{19773} - \tfrac{102500}{6591}, \\ f_6(x) &= f_4(x) - (-\tfrac{2313441}{5828750}x - \tfrac{1398919977}{1087178450})f_5 = \tfrac{1288744821}{543589225}. \end{aligned}$$

In $\mathbb{Z}[x]$, as f_1, f_2 are both primitive, in fact $\gcd(f_1, f_2) = 1$.

It is desirable to have an algorithm that computes the GCD of two polynomials in $R[x]$, while remaining in the same coefficient ring R, and without the severe coefficient growth seen in Example 1. We will examine several alternatives, leading up to the subresultant PRS algorithm which achieves both of these objectives while enjoying some other efficiency properties as well.

2 Preliminaries

Throughout, R will denote a unique factorization domain (UFD). We will assume that there is an algorithm to compute GCDs in R (which is clearly necessary if we wish to compute GCDs in $R[x]$).

Recall that the *content* of $0 \neq f \in R[x]$, denoted $\mathrm{cont}(f)$, is the GCD of its coefficients. We say that f is *primitive* if $\mathrm{cont}(f)$ is a unit. The *primitive part* of f is

$$\mathrm{prim}(f) = \frac{f}{\mathrm{cont}(f)}.$$

Two polynomials $f, g \in R[x]$ are *similar* if there exist $0 \neq a, b \in R$ with $af = bg$. For $f, g \neq 0$ this is equivalent to $\mathrm{prim}(f) = \mathrm{prim}(g)$.

Note that the factorization of f consists of the factorization of $\mathrm{cont}(f)$, together with the factorization of $\mathrm{prim}(f)$ which by Gauss's lemma will consist of other primitive polynomials. Thus to compute GCDs in $R[x]$, it suffices to consider the content and the primitive part separately:

$$\gcd(f, g) = \gcd(\mathrm{cont}(f), \mathrm{cont}(g)) \gcd(\mathrm{prim}(f), \mathrm{prim}(g)).$$

Since we (assume we) already have an algorithm to compute GCDs in R, we may focus on computing $\gcd(\mathrm{prim}(f), \mathrm{prim}(g))$.

Another way of interpreting this is that we may compute $\gcd(f, g)$ up to similarity, i.e. ignore constant factors, since if we can find h with $h \sim \gcd(f, g)$ then we will have

$$\gcd(f, g) = \gcd(\mathrm{cont}(f), \mathrm{cont}(g)) \, \mathrm{prim}(f).$$

In fact, this was implicitly done in Example 1 above when we worked in the fraction field of R. We can rephrase this as the most basic general GCD algorithm for $R[x]$:

Theorem 1 (Fractional Euclidean algorithm)**.** *Let $0 \neq f, g \in R[x]$, and let K be the fraction field of R. The following algorithm computes* $\gcd(f, g)$ *in $R[x]$.*

1. *Compute $h = \gcd(f, g)$ in $K[x]$ using the Euclidean algorithm.*
2. *Compute $c = \gcd(\mathrm{cont}(f), \mathrm{cont}(g)) \in R$.*
3. *Return $c \, \mathrm{prim}(h)$ (where $\mathrm{prim}(h)$ can be interpreted as e.g. first clearing denominators then computing the primitive part in $R[x]$).*

3 Pseudodivision

As mentioned in the introduction, the division algorithm cannot necessarily be carried out in $R[x]$ if R is not a field. However, the algorithm can be carried out if the divisor is monic. The process of *pseudodivision* can be seen as a generalization of this fact: it is a version of the division algorithm which effectively factors out possible denominators beforehand.

Lemma 1 (Pseudodivision)**.** *Let $f(x), g(x) \in R[x]$ with $g \neq 0$, and let b be the leading coefficient of g. Let $d = \deg f - \deg g + 1$ if $\deg f \geq \deg g$, and $d = 0$ otherwise. Then there exist unique $q(x), r(x) \in R[x]$ such that*

$$b^d f = gq + r, \qquad \deg r < \deg g.$$

Proof. By induction on $\deg f$. If $\deg f < \deg g$ then clearly $q = 0, r = f$ is the unique solution. Now suppose $\deg f \geq \deg g$ and the claim holds for all f of smaller degree. Let a be the leading coefficient of f. Then $\deg(bf - ag) < \deg f$, so by the induction hypothesis, there exist unique $s(x), r(x) \in R[x]$ such that

$$b^e(bf - ag) = gs + r, \qquad \deg r < \deg g$$

where e is $\deg(bf - ag) - \deg g + 1$ or 0. Since $\deg f \geq \deg g$, $d = \deg f - \deg g + 1 \geq 1$, and since also $\deg(bf - ag) < \deg f$, $d > e$. So rearranging gives

$$b^d f = g(b^{d-e-1}s + ab^{d-1}) + r,$$

so with $q = b^{d-e-1}s + ab^{d-1}$, we have existence. Uniqueness follows from uniqueness of s, r. □

Definition 1. If f, g are as in the above lemma, then q is the *pseudoquotient* and r is the *pseudoremainder* of f modulo g, and r is denoted $\operatorname{prem}(f, g)$.

Pseudodivision is just the usual division algorithm applied to $c^{\deg f - \deg g + 1} f$ and g, and the proof of the above lemma is directly analogous. Note that if R is a field (or if we work in the fraction field of R), then $\operatorname{prem}(f, g)$ is similar to the usual remainder of f modulo g. Since, as previously noted, GCD computation in $R[x]$ can be done up to similarity, we immediately have the following GCD algorithm:

Theorem 2 (Pseudo-Euclidean algorithm)**.** *Let $0 \neq f, g \in R[x]$. The following algorithm computes $\gcd(f, g)$ in $R[x]$.*

1. *Set $f_1 = f, f_2 = g$ and $k = 2$.*
2. *As long as $f_k \neq 0$, compute $f_{k+1} = \operatorname{prem}(f_{k-1}, f_k)$ and increment k.*
3. *Compute $c = \gcd(\operatorname{cont}(f), \operatorname{cont}(g)) \in R$.*
4. *Return $c \operatorname{prim}(f_{k-1})$.*

Proof. The proof is essentially the same as for the ordinary Euclidean algorithm. From the definition of pseudodivision we see that $\gcd(f,g) \mid \operatorname{prem}(f,g)$, so (by induction) $\gcd(f,g) \mid f_i$ for each i and so $\gcd(f,g) = c\operatorname{prim}(\gcd(f,g)) \mid c\operatorname{prim}(f_{k-1})$. Clearly $c \mid \gcd(f,g)$. From the definition of pseudodivision we see that $f_k = 0 \implies \operatorname{prim}(f_{k-1}) \mid f_{k-2}$, and then (by induction) $\operatorname{prim}(f_{k-1}) \mid f_i$ for all i, so $\operatorname{prim}(f_{k-1}) \mid f, g$ and so $\operatorname{prim}(f_{k-1}) \mid \gcd(f,g)$. □

Returning to Example 1 from the Introduction, we see that the pseudo-Euclidean algorithm, while remaining in the coefficient ring R, suffers from even worse coefficient growth.

Example 2. With

$$\begin{aligned} f_1(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5, \\ f_2(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21, \end{aligned}$$

the pseudo-Euclidean algorithm produces the following sequence of remainders:

$$\begin{aligned} f_3(x) &= -15x^4 + 3x^2 - 9, \\ f_4(x) &= 15795x^2 + 30375x - 59535, \\ f_5(x) &= 1254542875143750x - 1654608338437500, \\ f_6(x) &= 12593338795500743100931141992187500. \end{aligned}$$

4 Polynomial remainder sequences

The following definition captures the idea of a sequence of remainders computed by a variant of the Euclidean algorithm up to similarity.

Definition 2. Let $0 \neq f, g \in R[x]$. A *polynomial remainder sequence* (PRS) beginning with f, g is a sequence of polynomials

$$f_1 = f, f_2 = g, f_3, \ldots, f_k \in R'[x],$$

where R' is either R or its fraction field, such that

$$f_i \sim \operatorname{prem}(f_{i-2}, f_{i-1}) \qquad \text{for } i = 3, \ldots, k.$$

Thus, the fractional Euclidean algorithm and the pseudo-Euclidean algorithm are both instances of the following template.

Definition 3. Let $0 \neq f, g \in R[x]$. A *PRS algorithm* is an algorithm to compute $\gcd(f,g)$ of the following form:

1. Compute a PRS $f_1 = f, f_2 = g, \ldots, f_k = 0$.
2. Compute $c = \gcd(\operatorname{cont}(f), \operatorname{cont}(g))$.

3. Return $c\,\mathrm{prim}(f_{k-1})$.

Lemma 2. *If $f \sim f'$ and $0 \neq g \sim g'$ then $\mathrm{prem}(f,g) \sim \mathrm{prem}(f',g')$.*

Proof. If $\deg f = \deg f' < \deg g = \deg g'$ then $\mathrm{prem}(f,g) = f \sim f' = \mathrm{prem}(f',g')$, so assume otherwise and let $d = \deg f - \deg g + 1 = \deg f' - \deg g' + 1$. Let b, b' be the leading coefficients of g, g' respectively. Then there exist unique q, q' with

$$b^d f = gq + \mathrm{prem}(f,g), \qquad\qquad b'^d f' = g'q' + \mathrm{prem}(f',g').$$

Since $f \sim f'$ and $g \sim g'$, there exist $0 \neq u, u', v, v' \in R$ with $uf = u'f', vg = v'g', vb = v'b'$. The two equations above give

$$\begin{aligned}(vb)^d uf &= (vg)(uv^{d-1}q) + uv^d\,\mathrm{prem}(f,g),\\ (vb)^d uf = (v'b')^d u'f' &= (v'g')(u'v'^{d-1}q') + u'v'^d\,\mathrm{prem}(f,g)\\ &= (vg)(u'v'^{d-1}q') + u'v'^d\,\mathrm{prem}(f,g).\end{aligned}$$

Note that vb is the leading coefficient of vg. Thus by the uniqueness of pseudodivision (of uf modulo vg),

$$uv^{d-1}q = u'v'^{d-1}q', \qquad\qquad uv^d\,\mathrm{prem}(f,g) = u'v'^d\,\mathrm{prem}(f,g). \qquad \square$$

Corollary 1. *Every PRS $f_1 = f, f_2 = g, f_3, \ldots, f_k$ is termwise similar to the pseudo-Euclidean PRS $g_1 = f, g_2 = g, g_3 = \mathrm{prem}(g_1, g_2), \ldots, g_k = \mathrm{prem}(g_{k-2}, g_{k-1})$.*

Proof. By induction on k. For $k = 2$ this is trivial. If the claim holds for $k-1$, then $f_{k-2} \sim g_{k-2}$ and $f_{k-1} \sim g_{k-1}$, so using Lemma 2, $f_k \sim \mathrm{prem}(f_{k-2}, f_{k-1}) \sim \mathrm{prem}(g_{k-2}, g_{k-1}) = g_k$. $\square$

Corollary 2. *Every PRS algorithm correctly computes $\gcd(f,g)$.*

Proof. Follows from Corollary 1 and Theorem 2, since a PRS algorithm (by definition) yields an answer that only depends on the primitive part of the PRS. $\square$

We conclude this section by giving another "obvious" PRS algorithm. By inspection, we can see that all of the intermediate remainders in Example 2 have rather large contents. One idea is to simply remove the content from the pseudoremainder after each division.

Theorem 3 (Primitive PRS algorithm)**.** *Let $0 \neq f, g \in R[x]$. The following algorithm computes $\gcd(f,g)$ in $R[x]$.*

1. *Set $f_1 = f, f_2 = g$ and $k = 2$.*
2. *As long as $f_k \neq 0$, compute $f_{k+1} = \mathrm{prim}(\mathrm{prem}(f_{k-1}, f_k))$ and increment k.*
3. *Compute $c = \gcd(\mathrm{cont}(f), \mathrm{cont}(g)) \in R$.*
4. *Return $c\,\mathrm{prim}(f_{k-1})$.*

Proof. By Corollary 2 with the *primitive PRS*

$$f_1 = f, f_2 = g, f_3 = \text{prim}(\text{prem}(f_1, f_2)), \ldots, f_k = \text{prim}(\text{prem}(f_{k-2}, f_{k-1})) = 0. \qquad \square$$

Example 3. With

$$\begin{aligned} f_1(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5, \\ f_2(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21, \end{aligned}$$

the primitive PRS algorithm produces the following PRS:

$$\begin{aligned} f_3(x) &= 5x^4 - x^2 + 3, \\ f_4(x) &= 13x^2 + 25x - 49, \\ f_5(x) &= 4663x - 6150, \\ f_6(x) &= 1. \end{aligned}$$

This is a huge improvement over Examples 1 and 2. The tradeoff is the need to compute the content of each pseudoremainder in order to arrive at the primitive part. This is hidden in the example as shown above (since only the primitive parts, which make up the PRS, are shown), but for example, f_5 is computed as

$$f_5(x) = \text{prim}(-233150x + 307500).$$

With higher-degree polynomials requiring a longer PRS, or with R a UFD in which computing GCDs is expensive (e.g. R is itself a polynomial ring), it would be desirable not to have to compute the content after each step, especially since the final nonzero term in the PRS is likely to have low degree and thus a more easily computable content.

5 Subresultants

Throughout this section, let $R[x]^n \subset R[x]$ denote polynomials of degree at most n, let the *standard basis* for $R[x]^n$ be $x^n, \ldots, 1$ (in that order), and let the standard basis for $R[x]^n/R[x]^m$, $m \leq n$, be $x^n, \ldots, x^{m+1}$. Fix $f \in R[x]^m$ and $g \in R[x]^n$ and set

$$f(x) = \sum_{i=0}^{m} a_i x^i, \qquad\qquad g(x) = \sum_{i=0}^{n} b_i x^i.$$

Recall that the *resultant* of f and g, $\text{Res}(f, g)$, can be defined as follows: the map

$$\begin{aligned} \sigma(f,g) : R[x]^{n-1} \times R[x]^{m-1} &\longrightarrow R[x]^{n+m-1} \\ (u, v) &\longmapsto uf + vg \end{aligned}$$

has $(n+m)\times(n+m)$ matrix with respect to the standard bases

$$S(f,g)=\begin{bmatrix} a_m & & & b_n & & \\ a_{m-1} & \ddots & & b_{n-1} & \ddots & \\ \vdots & \ddots & a_m & \vdots & \ddots & b_n \\ a_0 & \ddots & a_{m-1} & b_0 & \ddots & b_{n-1} \\ & \ddots & \vdots & & \ddots & \vdots \\ & & a_0 & & & b_0 \end{bmatrix};$$

the resultant is

$$\mathrm{Res}(f,g)=\det S(f,g),$$

so that $\mathrm{Res}(f,g)=0$ iff there are $u\in R[x]^{n-1}, v\in R[x]^{m-1}$, not both zero, with $uf+vg=0$. The *subresultants* are defined via a generalization of this procedure.

Definition 4. For $j\le m,n$, not both equal, consider the map

$$\begin{aligned}\sigma_j(f,g): R[x]^{n-j-1}\times R[x]^{m-j-1} &\longrightarrow R[x]^{n+m-j-1}\\ (u,v) &\longmapsto uf+vg\end{aligned}$$

and call its $(n+m-j)\times(n+m-2j)$ matrix $S_j(f,g)$. (Thus $S_j(f,g)$ looks like $S(f,g)$ above, except with columns $1,\dots,j$ and $n+1,\dots,n+j$ and rows $1,\dots,j$ deleted.) Compose this map with the projection

$$\pi_j: R[x]^{n+m-j-1}\longrightarrow R[x]^{n+m-j-1}/R[x]^{j-1}$$

and call the $(n+m-2j)\times(n+m-2j)$ matrix of $\pi_j\circ\sigma_j(f,g)$ by $\hat{S}_j(f,g)$; thus $\hat{S}_j(f,g)$ is $S_j(f,g)$ with the bottom j rows deleted:

$$\hat{S}_j(f,g)=\begin{bmatrix} a_m & & & b_n & & \\ a_{m-1} & \ddots & & b_{n-1} & \ddots & \\ \vdots & \ddots & a_m & \vdots & \ddots & b_n \\ a_0 & \ddots & \vdots & b_0 & \ddots & \vdots \\ & \ddots & a_j & & \ddots & b_j \end{bmatrix}.$$

The *jth scalar subresultant* of f and g is

$$\det\hat{S}_j(f,g),$$

and the *jth (polynomial) subresultant* of f and g is the polynomial

$$\mathrm{Res}_j(f,g)=\begin{cases}(\det\hat{S}_j(f,g))\sigma_j(f,g)((\pi_j\circ\sigma_j(f,g))^{-1}(x^j)) & \text{if } \det\hat{S}_j(f,g)\neq 0,\\ 0 & \text{otherwise.}\end{cases}$$

The scalar subresultant $\det \hat{S}_j(f,g)$ is zero iff $uf + vg$ has degree less than j for some $u \in R[x]^{n-j-1}, v \in R[x]^{m-j-1}$ not both zero, and the polynomial subresultant $\mathrm{Res}_j(f,g)$ is the unique polynomial of the form $uf + vg$ with degree at most j and coefficient $\det \hat{S}_j(f,g)$ of x^j. This can be seen more clearly via the matrix representation. Let $S_j^{[i]}(f,g)$ be the $(n+m-2j) \times (n+m-2j)$ submatrix of $S_j(f,g)$ consisting of the top $n+m-2j-1$ rows and the $(n+2m-j-i)$th row; thus $\hat{S}_j(f,g) = S_j^{[j]}(f,g)$. The matrix of $\mathrm{Res}_j(f,g)$ is

$$(\det \hat{S}_j(f,g)) S_j(f,g) (\hat{S}_j(f,g))^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = S_j(f,g)(\operatorname{adj} \hat{S}_j(f,g)) \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \det S_j^{[j]}(f,g) \\ \vdots \\ \det S_j^{[0]}(f,g) \end{bmatrix}.$$

In other words, we have the explicit formula

$$\mathrm{Res}_j(f,g) = \sum_{i=0}^{j} (\det S_j^{[i]}(f,g)) x^i \in R[x]^j. \tag{1}$$

This is the origin of the name "subresultant" [2]: the coefficients are determinants of submatrices of the Sylvester matrix $S(f,g)$.

The notations $\sigma_j, S_j, \pi_j, \hat{S}_j, \mathrm{Res}_j$ are somewhat misleading, because they do not encode the numbers m, n for which we are considering $f \in R[x]^m$ and $g \in R[x]^n$. While we will usually take m, n to be the degrees of f, g, it is convenient to allow $\deg f \le m, \deg g \le n$; the definitions above still make sense. To indicate the choice of m, n, we will sometimes write

$$\sigma_j^{m,n}, \qquad S_j^{m,n}, \qquad \pi_j^{m,n}, \qquad \hat{S}_j^{m,n}, \qquad \mathrm{Res}_j^{m,n}.$$

Note that for $j = 0$, we have

$$\sigma_0 = \sigma, \qquad S_0(f,g) = \hat{S}_0(f,g) = S(f,g), \qquad \mathrm{Res}_0(f,g) = \det \hat{S}_0(f,g) = \mathrm{Res}(f,g).$$

Since $\mathrm{Res}_0(f,g) = uf + vg$ for some $u \in R[x]^n, v \in R[x]^m$ by definition, we see again that $\mathrm{Res}(f,g) = 0$ iff $uf + vg = 0$ has a nontrivial solution.

Lemma 3 (Properties of subresultants)**.**

(a) $\mathrm{Res}_j^{m,n}(f,g) = (-1)^{(m-j)(n-j)} \mathrm{Res}_j^{n,m}(g,f)$.

(b) $\mathrm{Res}_j(af, bg) = a^{n-j} b^{m-j} \mathrm{Res}_j(f,g)$ *for all* $a, b \in R$.

(c) $\mathrm{Res}_j^{m,j}(f,g) = b_j^{m-j-1} g$, *and if* $b_n = 0$ *and* $n \ge k \ge j$, *then* $\mathrm{Res}_j^{m,n}(f,g) = a_m^{n-k} \mathrm{Res}_j^{m,k}(f,g)$. *In particular, if* $a_m = b_n = 0$ *and either* $m > j$ *or* $n > j$, *then* $\mathrm{Res}_j^{m,n}(f,g) = 0$.

(d) $\mathrm{Res}_j^{m,n}(f,g) = \mathrm{Res}_j^{m,n}(f + gh, g)$ *for all* $h \in R[x]^{m-n}$.

Proof. (a) and (b) are immediate from the explicit formula (1). For (c), repeatedly expand along the top row in each $S_j^{[i]}(f,g)$; for $n = j$, $S_j^{[i]}(f,g)$ is lower triangular with all diagonal entries b_j except the last, which is b_i, yielding

$$\det S_j^{[i]}(f,g) = b_j^{m-j-1} b_i,$$

so by the explicit formula, $\operatorname{Res}_j^{m,j}(f,g) = b_j^{m-j-1} g$. For (d), note that

$$\sigma_j(f+gh,g)(u,v) = u(f+gh) + vg = uf + (uh+v)g = \sigma_j(f,g)(u, uh+v).$$

Since $\deg h \le m-n$ and $\deg u < n-j$, $\deg(uh+v) < m-j$, so the map $(u,v) \mapsto (u, uh+v)$ is an endomorphism of $R[x]^{n-j-1} \times R[x]^{m-j-1}$; its determinant is clearly 1 (e.g. its matrix is a triangular block matrix with identity diagonal blocks), so its composition with $\sigma_j(f,g)$ does not change any of the relevant determinants, and so $\operatorname{Res}_j^{m,n}(f+gh,g) = \operatorname{Res}_j^{m,n}(f,g)$. □

6 The subresultant PRS

Part (d) of Lemma 3 above is crucial, for it tells us that subresultants behave nicely with respect to (pseudo)remainders.

Lemma 4. *Suppose* $\deg f = m \ge \deg g = n$, *and* $q, r \in R[x]$ *are such that*

$$f = gq + r, \qquad \deg r < n = \deg g.$$

Let $l = \deg r$ *and* $r = \sum_{i=0}^{l} c_i x^i$. *Then*

$$\begin{aligned}
\operatorname{Res}_j(f,g) &= (-1)^{(m-j)(n-j)} b_n^{m-l} \operatorname{Res}_j(g,r) && 0 \le j < l,\\
\operatorname{Res}_l(f,g) &= (-1)^{(m-l)(n-l)} b_n^{m-l} c_l^{n-l-1} r, &&\\
\operatorname{Res}_j(f,g) &= 0 && l < j < n-1,\\
\operatorname{Res}_{n-1}(f,g) &= (-1)^{m-n+1} b_n^{m-n+1} r. &&
\end{aligned}$$

Here $\operatorname{Res}_j(g,r)$ *denotes* $\operatorname{Res}_j^{n,l}(g,r)$.

Proof. By Lemma 3(d,a,c),

$$\begin{aligned}
\operatorname{Res}_j(f,g) &= \operatorname{Res}_j^{m,n}(r,g)\\
&= (-1)^{(m-j)(n-j)} \operatorname{Res}_j^{n,m}(g,r)\\
&= \begin{cases} (-1)^{(m-j)(n-j)} b_n^{m-l} \operatorname{Res}_j^{n,l}(g,r) & \text{if } j < l,\\ (-1)^{(m-j)(n-j)} b_n^{m-j} c_j^{n-j-1} r & \text{if } l \le j; \end{cases}
\end{aligned}$$

for $l < j$, $c_j = 0$ since r is degree l, so all four cases follow. □

Corollary 3. *Suppose* $\deg f = m \geq \deg g = n$*, and let* $c_l x^l$ *be the leading term in* $\operatorname{prem}(f,g)$*. Then*

$$\begin{aligned} b_n^{(m-n+1)(n-j)} \operatorname{Res}_j(f,g) &= (-1)^{(m-j)(n-j)} b_n^{m-l} \operatorname{Res}_j(g, \operatorname{prem}(f,g)) && 0 \leq j < l, \\ b_n^{(m-n+1)(n-l)} \operatorname{Res}_l(f,g) &= (-1)^{(m-l)(n-l)} b_n^{m-l} c_l^{n-l-1} \operatorname{prem}(f,g), \\ \operatorname{Res}_j(f,g) &= 0 && l < j < n-1, \\ \operatorname{Res}_{n-1}(f,g) &= (-1)^{m-n+1} \operatorname{prem}(f,g). \end{aligned}$$

Proof. Replace f with $b_n^{m-n+1} f$ and r with $\operatorname{prem}(f,g)$ in Lemma 4 and use Lemma 3(b). □

Corollary 4. *Suppose* $\deg f = m \geq \deg g = n$*, and let* $f_1 = f, f_2 = g, f_3, \ldots, f_k = 0$ *be a PRS. Then for* $i = 3, \ldots, k$*,*

$$\operatorname{Res}_{\deg f_i}(f,g) \sim \operatorname{Res}_{\deg f_{i-1}-1}(f,g) \sim f_i.$$

Furthermore, $\operatorname{Res}_j(f,g) = 0$ *for all other* $j < n$*.*

Proof. By Lemma 3(b), if $g \sim h$ then $\operatorname{Res}_j(f,g) \sim \operatorname{Res}_j(f,h)$. Thus by Corollary 3, for $\deg f_i \leq j \leq \deg f_{i-1} - 1$,

$$\begin{aligned} \operatorname{Res}_j(f,g) &\sim \operatorname{Res}_j(g, \operatorname{prem}(f,g)) \sim \operatorname{Res}_j(g, f_3) \sim \cdots \sim \operatorname{Res}_j(f_{i-2}, f_{i-1}) \\ &\sim \begin{cases} \operatorname{prem}(f_{i-2}, f_{i-1}) \sim f_i & \text{if } j = \deg f_i \text{ or } \deg f_{i-1} - 1, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Since $f_k = 0$, this covers all $j < n$. □

Thus, the subresultants of f, g contain essentially the same information as a PRS. The *subresultant PRS* is the PRS for which the second similarity in Corollary 4 becomes equality.

Definition 5. The *subresultant PRS* of f, g is the PRS

$$f_1 = f, f_2 = g, f_3 = \operatorname{Res}_{\deg f_2 - 1}(f,g), \ldots, f_k = \operatorname{Res}_{\deg f_{k-1}-1}(f,g) = 0.$$

From the definition of subresultants, we get for free that the subresultant PRS lies in $R[x]$. Corollary 4 tells us that the subresultant PRS is a PRS; the next lemma shows how to compute the subresultant PRS, and can be derived by carefully tracing through the proof of Corollary 4 while keeping track of the constant factors hidden in similarity relations.

Lemma 5. *Let* $f, g \in R[x]$ *and let* $f_1 = f, f_2 = g, f_3, \ldots, f_k = 0$ *be the subresultant PRS of* f, g*. Let* $n_i = \deg f_i$ *for* $i = 1, \ldots, k-1$*, let* $a_1 = 1$*, and let* a_i *be the leading coefficient* f_i *for* $i = 2, \ldots, k-1$*. Then*

$$f_i = \frac{(-1)^{n_{i-2} - n_{i-1} + 1} \operatorname{prem}(f_{i-2}, f_{i-1})}{a_{i-2} c_{i-2}^{n_{i-2} - n_{i-1}}} \qquad i = 3, \ldots, k,$$

where

$$c_1 = 1, \qquad c_i = a_i^{n_{i-1} - n_i} c_{i-1}^{n_i - n_{i-1} + 1} \qquad i = 2, \ldots, k-2.$$

The reader is warned that the following proof is tedious and not particularly enlightening.

Proof. By induction on i. The claim for $i = 3$ follows directly from Corollary 3. Now suppose $i \geq 4$ and the claim holds for all smaller i. By Corollary 3, for all $3 \leq k \leq i-1$,

$$\begin{aligned} &a_{k-1}^{(n_{k-2}-n_{k-1}+1)(n_{k-1}-n_{i-1}+1)} \operatorname{Res}_{n_{i-1}-1}(f_{k-2}, f_{k-1}) \\ &\qquad = (-1)^{(n_{k-2}-n_{i-1}+1)(n_{k-1}-n_{i-1}+1)} a_{k-1}^{n_{k-2}-n_k} \operatorname{Res}_{n_{i-1}-1}(f_{k-1}, \operatorname{prem}(f_{k-2}, f_{k-1})). \end{aligned}$$

Substituting in the induction hypothesis

$$\operatorname{prem}(f_{k-2}, f_{k-1}) = (-1)^{n_{k-2}-n_{k-1}+1} a_{k-2} c_{k-2}^{n_{k-2}-n_{k-1}} f_k$$

and simplifying using Lemma 3(b) gives

$$\begin{aligned} &a_{k-1}^{(n_{k-2}-n_{k-1}+1)(n_{k-1}-n_{i-1}+1)} \operatorname{Res}_{n_{i-1}-1}(f_{k-2}, f_{k-1}) \\ &\qquad = a_{k-1}^{n_{k-2}-n_k} a_{k-2}^{n_{k-1}-n_{i-1}+1} c_{k-2}^{(n_{k-2}-n_{k-1})(n_{k-1}-n_{i-1}+1)} \operatorname{Res}_{n_{i-1}-1}(f_{k-1}, f_k), \end{aligned}$$

then multiplying by $c_{k-1}^{(n_{k-1}-n_{k-2}+1)(n_{k-1}-n_{i-1}+1)}$ and using $c_{k-1} = a_{k-1}^{n_{k-2}-n_{k-1}} c_{k-2}^{n_{k-1}-n_{k-2}+1}$ gives

$$\begin{aligned} &a_{k-1}^{n_{k-1}-n_{i-1}+1} c_{k-1}^{n_{k-1}-n_{i-1}+1} \operatorname{Res}_{n_{i-1}-1}(f_{k-2}, f_{k-1}) \\ &\qquad = a_{k-1}^{n_{k-2}-n_k} a_{k-2}^{n_{k-1}-n_{i-1}+1} c_{k-2}^{n_{k-1}-n_{i-1}+1} \operatorname{Res}_{n_{i-1}-1}(f_{k-1}, f_k), \end{aligned}$$

and multiplying by $a_{k-1}^{n_k-n_{k-1}}$, using the same identity, and cancelling c_{k-1} gives

$$a_{k-1}^{n_k-n_{i-1}+1} c_{k-1}^{n_{k-1}-n_{i-1}} \operatorname{Res}_{n_{i-1}-1}(f_{k-2}, f_{k-1}) = a_{k-2}^{n_{k-1}-n_{i-1}+1} c_{k-2}^{n_{k-2}-n_{i-1}} \operatorname{Res}_{n_{i-1}-1}(f_{k-1}, f_k).$$

Now cross-comparing values for $k = 3, \ldots, i-1$ gives

$$a_{i-2} c_{i-2}^{n_{i-2}-n_{i-1}} \operatorname{Res}_{n_{i-1}-1}(f, g) = a_1^{n_2-n_{i-1}+1} c_1^{n_1-n_{i-1}} \operatorname{Res}_{n_{i-1}-1}(f_{i-2}, f_{i-1}),$$

and since $a_1 = c_1 = 1$, by Corollary 3,

$$f_i = \operatorname{Res}_{n_{i-1}-1}(f, g) = \frac{\operatorname{Res}_{n_{i-1}-1}(f_{i-2}, f_{i-1})}{a_{i-2} c_{i-2}^{n_{i-2}-n_{i-1}}} = \frac{(-1)^{n_{i-2}-n_{i-1}+1} \operatorname{prem}(f_{i-2}, f_{i-1})}{a_{i-2} c_{i-2}^{n_{i-2}-n_{i-1}}}. \qquad \square$$

Thus, we have the subresultant PRS algorithm, which computes $\gcd(f, g)$ while remaining in $R[x]$ and removing part of the content at each step without computing coefficient GCDs:

Theorem 4 (Subresultant PRS algorithm)**.** *Let $0 \neq f, g \in R[x]$. The following algorithm computes* $\gcd(f, g)$ *in* $R[x]$.

1. *Set $f_1 = f, f_2 = g$ and $k = 2$.*
2. *Compute the subresultant PRS $f_1 = f, f_2 = g, \ldots, f_k = 0$ recursively using Lemma 5.*
3. *Compute $c = \gcd(\operatorname{cont}(f), \operatorname{cont}(g)) \in R$.*

4. *Return* $c\,\mathrm{prim}(f_{k-1})$.

Proof. By Corollaries 2 and 4. □

We note that in order to compute $\gcd(f, g)$, it is unnecessary (although easy) to keep track of the signs in Lemma 5, and the following example does not do so.

Example 4. With

$$f_1(x) = x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5,$$
$$f_2(x) = 3x^6 + 5x^4 - 4x^2 - 9x + 21,$$

the subresultant PRS algorithm (ignoring signs) produces the following PRS:

$$f_3(x) = -15x^4 + 3x^2 - 9,$$
$$f_4(x) = 65x^2 + 125x - 245,$$
$$f_5(x) = -9326x + 12300,$$
$$f_6(x) = 260708.$$

While the coefficients are not as small as in the primitive PRS (Example 3), they are much smaller than those in both the fractional Euclidean PRS (Example 1) and the pseudo-Euclidean PRS (Example 2). Furthermore, no GCDs need to be computed in R in order to build the PRS.

References

[1] Brown, W. S., "On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors", *Journal of the Association for Computing Machinery*, pp. 478-504, 18 (4): 1971.

[2] Brown, W. S. and Traub, J. F., "On Euclid's Algorithm and the Theory of Subresultants", *Journal of the Association for Computing Machinery*, pp. 505-514, 18 (4): 1971.

[3] Brown, W. S., "The Subresultant PRS Algorithm", *ACM Transactions on Mathematical Software*, 4 (3) : 1978.

[4] Knuth, D. E., *The Art of Computer Programming*, vol. 2, Addison-Wesley: 1969 (cited in [1, 6]).

[5] Loos, R., "Generalized Polynomial Remainder Sequences", edited by Buchberger, B. et al., *Computer Algebra*, Springer-Verlag: 1983.

[6] Zippel, R., *Effective Polynomial Computation*, Kluwer Academic Publishers: 1993.