Mathematics and Algorithms for Computer Algebra

July 9, 2003

5: Polynomial GCDs and remainder sequences

The first and last sections are applications of gcds; the intermediate sections are about how to compute polynomial gcds.

1 Squarefree decomposition

This provides a restricted factorization of a polynomial and so could also be called squarefree factorization, except that factorization usually implies factorization into irreducible factors, which a squarefree decomposition is not. Squarefree decomposition is much simpler and therefore faster than full factorization and requires only formal derivatives and gcds. It is an obvious precursor to a full factorization, but there are many situations in which a squarefree decomposition is sufficient – for example, to determine whether a polynomial is a perfect n^{th} power, and if so to find its n^{th} root. This section is based on Mignotte.

1.1 The basic theory

Let F be a field, either infinite or finite, which requires a slightly more general formulation than is required for the infinite case alone. The infinite field case can trivially be generalized to an integral domain by dropping the condition that polynomials be monic.

Definition 1 $f \in F[x]$ *is* squarefree *if it is not divisible by the square of any non-constant polynomial over F, or equivalently if f has only simple roots in any field that contains F.*

Note that squarefree does not imply irreducible, although irreducible does imply squarefree. For example, over $\mathbb{Q}$:

- $x^2 + 3x + 1$ is squarefree and irreducible;
- $x^2 + 3x + 2 = (x + 1)(x + 2)$ is squarefree but not irreducible;
- $x^2 + 2x + 1 = (x + 1)^2$ is neither squarefree nor irreducible.

Let f' be the (*formal*) derivative of f and let $g = \gcd(f, f')$ be their *monic* gcd. If $g = 1$ then there does not exist any non-constant polynomial that divides both f and f', and hence f is squarefree. Conversely, if f is not squarefree and possesses, over some field $E \geq F$, the decomposition

$$f = \lambda m_1^{e_1} \cdots m_k^{e_k}, \quad e_1 \geq 2, \ e_{i \neq 1} \geq 1,$$

where the m_i are distinct monic polynomials over E, then the derivative

$$f' = \lambda m_1^{e_1 - 1} \cdots m_k^{e_k - 1} s \ \text{ where } \ s = \sum_{i=1}^{k} e_i m_1 \cdots m_{i-1} m_{i+1} \cdots m_k.$$

Therefore, the polynomial

$$m_1^{e_1 - 1} \cdots m_k^{e_k - 1} \neq 1$$

because $e_1 \geq 2$ and it divides $g = \gcd(f, f')$, so f, f' are not coprime. Moreover, the quotient f/g divides the product $m_1 \cdots m_k$ and therefore f/g is squarefree.

Hence we have proved

Proposition 1 *Let f be a non-constant polynomial, with coefficients in some field F. Then f is squarefree if and only if f and its derivative f' are relatively prime. Moreover, the polynomial $h = f/\gcd(f, f')$ is always squarefree, and if $f' \neq 0$ and $\gcd(f, f') \neq 1$ then h is a non-trivial factor of f.*

1.2 Squarefree decomposition in characteristic zero

If F has characteristic zero then none of the e_i in s can vanish and hence $f' \neq 0$ and no m_i divides s. Therefore

$$g = \gcd(f, f') = m_1^{e_1 - 1} \cdots m_k^{e_k - 1} \neq 1$$

because $e_1 \geq 2$, and

$$h = f/g = \lambda m_1 \cdots m_k$$

is a squarefree polynomial, which is the product of all the distinct squarefree factors of f.

The aim of squarefree decomposition over a field is to compute a set of monic coprime polynomials m_i such that the polynomial f, now assumed for convenience to be monic, has the decomposition

$$f = m_1 m_2^2 \cdots m_k^k, \quad k \geq 1.$$

If no squarefree factor occurs raised to the power i then $m_i = 1$ and so can be omitted from the product. It is trivial to make f monic by dividing by its leading coefficient if necessary.

1.3 Algorithms for characteristic zero

Yun[1] has analysed this problem is detail, and gives three algorithms. The first is due to Tobey and Horowitz:

input: monic $f \in F[x]$
$g_1 := \gcd(f, f')$; $h_1 := f/g_1$; $i := 1$;
while $g_i \neq 1$ do
begin
    $g_{i+1} := \gcd(g_i, g_i')$; $h_{i+1} := g_i/g_{i+1}$;
    $m_i := h_i/h_{i+1}$; $i := i + 1$
end.
output: monic $m_i \in F[x]$ such that $f = m_1 m_2^2 \cdots m_k^k$

It works as follows. If

$$f = m_1 m_2^2 m_3^3 \cdots m_k^k$$

then

$$g_1 = m_2 m_3^2 \cdots m_k^{k-1}, \quad h_1 = m_1 m_2 \cdots m_k,$$

where h_1 is "the squarefree part" of f. For $i > 1$,

$$g_i = m_{i+1} m_{i+2}^2 \cdots m_k^{k-i}, \quad h_i = m_i m_{i+1} m_{i+2} \cdots m_k,$$

and hence as claimed

$$h_i/h_{i+1} = m_i.$$

[1] D. Y. Y. Yun, "On squarefree decomposition algorithms", in *Proc. 1976 AMS Symp. on Symbolic and Algebraic Computing*, Yorktown Heights, NY, ed. R. D. Jenks.

This algorithm literally repeats the same basic algorithm for extracting the squarefree part. However, the repeated differentiation is not necessary, and is avoided in the following modification due to D. Musser, which leads to exactly the same intermediate variable values as the previous algorithm:

input: monic $f \in F[x]$
$g_1 := \gcd(f, f')$; $h_1 := f/g_1$; $i := 1$;
while $h_i \neq 1$ do
begin
$h_{i+1} := \gcd(g_i, h_i)$; $g_{i+1} := g_i/h_{i+1}$;
$m_i := h_i/h_{i+1}$; $i := i + 1$
end.
output: monic $m_i \in F[x]$ such that $f = m_1 m_2^2 \cdots m_k^k$

Yun shows that Musser's algorithm is more efficient than the first, and also gives another algorithm of his own, which is more complicated but also more efficient than Musser's.

1.4 Squarefree decomposition in nonzero characteristic

Mignotte shows that none of the three algorithms referred to above works correctly in nonzero characteristic, for which everything is more complicated. Suppose that F has characteristic p, and that as before f possesses the decomposition

$$f = \lambda m_1^{e_1} \cdots m_k^{e_k}, \quad e_1 \geq 2, \; e_{i \neq 1} \geq 1,$$

and hence

$$f' = \lambda m_1^{e_1 - 1} \cdots m_k^{e_k - 1} s \quad \text{where} \quad s = \sum_{i=1}^{k} e_i m_1 \cdots m_{i-1} m_{i+1} \cdots m_k.$$

Then a polynomial m_i divides s if and only if p divides e_i, so that the term of the sum s in which the factor m_i would be absent is itself absent.

If p does not divide all the exponents e_i then suppose that it divides only the first h for $0 \leq h < k$. Then $f' \neq 0$,

$$g = \gcd(f, f') = m_1^{e_1} \cdots m_h^{e_h} m_{h+1}^{e_{h+1} - 1} \cdots m_k^{e_k - 1}$$

and

$$f/g = \lambda m_{h+1} \cdots m_k.$$

Alternatively, p divides all the exponents e_i if and only if $f' = 0$. If $f = \sum f_i x^i$ and $f' = \sum i f_i x^{i-1} = 0$ then p divides each i for which $f_i \neq 0$. Let p^e be the greatest power of p that divides the gcd of all the i such that $f_i \neq 0$. Then $e \geq 1$ and f can be written as a composition of the form

$$f = H(x^{p^e}), \quad H \in F[x].$$

Conversely, if f has such a form then clearly $f' = 0$.

Suppose that F is the Galois Field $\mathbb{F}_{p^n} = GF(p^n)$. (Any finite field of characteristic p must be isomorphic to $\mathbb{F}_{p^n}$ for some $n \in \mathbb{Z}^+$.) Clearly p^e and $p^n - 1$ are coprime, so

$$\exists u, v, \quad up^e + v(p^n - 1) = 1, \quad 1 \leq u < p^n.$$

Now recall that $\mathbb{F}^*_{p^n}$ is a cyclic group of order $p^n - 1$. Then for every $a \in \mathbb{F}_{p^n}$ we have that either $a = 0$ or $a^{p^n-1} = 1$, and hence

$$(a^u)^{p^e} = a^{1-v(p^n-1)} = a,$$

i.e. a^u is a root of order p^e of a. In other words, we can solve the equation $A^{p^e} = a$ as $A = a^u$.

Recall that in characteristic p, for any $e \in \mathbb{N}$,

$$(a + b)^{p^e} = a^{p^e} + b^{p^e}$$

and hence inductively

$$(a + b + c + \cdots)^{p^e} = a^{p^e} + b^{p^e} + c^{p^e} + \cdots.$$

If $H_1(x) = \sum A_i x^i$ then

$$H_1(x)^{p^e} = \left(\sum A_i x^i\right)^{p^e} = \sum A_i^{p^e} (x^i)^{p^e}.$$

Therefore,

$$f(x) = H(x^{p^e}) = \sum a_i (x^{p^e})^i = H_1(x)^{p^e}$$

where

$$A_i = a_i^u,$$

and f is clearly neither squarefree nor irreducible.

This last computation is a good opportunity to apply the binary method for computing powers, in which case it requires at most $2 \lg u < 2n \lg p$

multiplications since $u < p^n$. Hence the computation of H_1 from H requires at most $O(\deg H.n \lg p)$ multiplications[2] in the finite field $\mathbb{F}_{p^n}$.

Proposition 1 together with the above analysis proves the following:

Proposition 2 *Every irreducible polynomial over a finite field or a field of characteristic zero is squarefree and therefore has nonzero derivative.*

Note that this does not apply over an infinite field with finite characteristic. For example, if $f(x) = yx^p - 1$ is a polynomial with coefficients in $\mathbb{F}_p(y)$, the field of rational functions in y with coefficients in the finite field of p elements, then $f' = 0$ but f is nevertheless irreducible. To see this, observe that in a suitable extension field, such as $\mathbb{C}(y^{1/p})$, f has the linear factorization

$$f(x) = \prod_{i=1}^{p} (y^{1/p}x - \omega^i) \quad \text{where} \quad \omega^p = 1,$$

but the product of any proper subset of these factors contains at least one coefficient that is a non-integer power of y and so $\notin \mathbb{F}_p(y)$.

1.5 A more general squarefree decomposition algorithm

This algorithm, given by Mignotte, works over a field of characteristic zero or a finite field of characteristic p. Let $f \in F[x]$ be a non-constant polynomial and f' its (formal) derivative.

If $f' \neq 0$ **then** compute $g = \gcd(f, f')$. If $g = 1$ then f is squarefree, otherwise $f = gh$ where h is squarefree and $0 < \deg h < \deg f$. Now apply the algorithm to g. [This is precisely the Tobey-Horowitz algorithm.]

If $f' = 0$ **then** compute $g \in \mathbb{F}_{p^n}[x]$ such that

$$f = g^{p^e}, \quad e \geq 1, \; g' \neq 0$$

as described above – this can happen only in finite characteristic. Now apply the algorithm to g. [It is in this case that the algorithms described by Yun all fail, because they assume $f' \neq 0$.]

[2] Mignotte gives this bound as $O(\deg H.p^e n \ln p)$ without explanation, and I do not see why.

2 GCDs in unique factorization domains

Polynomials over a field form a Euclidean domain, in which Euclid's algorithm using polynomial division can be used to compute polynomial gcds in complete analogy with the computation of integer gcds. However, neither polynomials over the integers nor multivariate polynomials over a field form Euclidean domains, although they form integral domains and hence gcds are defined. The question is how to compute gcds in these domains. It is not sufficient to simply compute in the field of fractions of the coefficient domain because, for example,

$$\gcd(6x^2y, 15xy^2 + 21x^3y^2) = \begin{cases} 3xy \text{ in } \mathbb{Z}[x,y], \\ xy \text{ in } \mathbb{Q}[x,y] \text{ because } 3 \text{ is a unit}, \\ x \text{ in } \mathbb{Q}(y)[x] \text{ because } 3y \text{ is a unit}. \end{cases}$$

The general situation that I want to consider is that the coefficients are elements of a *unique factorization domain* (UFD), which as I defined earlier is an integral domain in which every nonzero element is either a unit or has a representation as a product of primes that is unique up to order and factors of units. A nonunit element p is prime if $p = rq \Rightarrow$ either q or r is a unit. A field is a trivial UFD in which every nonzero element is a unit and there are no primes.

Proposition 3 *Polynomials over a UFD form a UFD, and hence multivariate polynomials over a UFD form a UFD.*

A polynomial that is "prime" is usually called "irreducible". A set of elements of a UFD is called *relatively prime* if no prime of that UFD divides all of them.

2.1 Primitive polynomials

Definition 2 *A polynomial over a UFD is called* primitive *if its coefficients are relatively prime.*

The following lemma due to Gauss is quoted in different forms by different authors, but one version implies another.

Lemma 4 (Gauss's Lemma) *The product of primitive polynomials over a UFD is primitive.*

Proof Let $u(x) = u_m x^m + \cdots + u_0$ and $v(x) = v_n x^n + \cdots + v_0$ be primitive polynomials. If p is any prime element of the UFD we need to prove that it does not divide all coefficients of $u(x)v(x)$. Because u, v are primitive there are indices i, j such that p does not divide u_i, v_j. Choose i, j to be the smallest possible such indices, so that p does divide $u_{i'<i}$ and $v_{j'<j}$. Then the coefficient of x^{i+j} in $u(x)v(x)$ is $u_0 v_{j+i} + u_1 v_{j+i-1} + \cdots + u_{i-1} v_{j+1} + u_i v_j + u_{i+1} v_{j-1} + \cdots + u_{i+j-1} v_1 + u_{i+j} v_0$ which is not divisible by p because $u_i v_j$ is not, but all the other terms are. □

If a nonzero polynomial over a UFD is not primitive, then primes common to all the coefficients can be successively factored out until the polynomial factor is primitive; in fact, this will factor out the gcd of the coefficients, which is unique up to a unit. This leads to the following

Lemma 5 *Any nonzero polynomial $u(x)$ over a UFD S can be factored in the form $u(x) = cv(x), c \in S$ where $v(x)$ is primitive. This factorization is unique up to a factor of a unit.*

Proof Existence is proved by the argument preceding the Lemma. To prove uniqueness, assume to the contrary that $c_1 v_1(x) = c_2 v_2(x)$, where $v_1(x)$ and $v_2(x)$ are primitive and c_1 is not a unit multiple of c_2. Then, by unique factorization, there exists a prime $p \in S$ and $k \in \mathbb{Z}^+$ such that p^k divides c_1 but not c_2. Then p^k must divide all the coefficients of $v_2(x)$, contradicting the assumption that $v_2(x)$ is primitive. □

Therefore, any nonzero polynomial $u(x)$ can be decomposed essentially uniquely as

$$u(x) = \mathrm{cont}(u) \cdot \mathrm{pp}(u(x)),$$

where $\mathrm{cont}(u) \in S$ is the "content", and $\mathrm{pp}(u(x))$, a primitive polynomial over S, is the "primitive part". When $u(x) = 0$ it is convenient to define $\mathrm{cont}(u) = \mathrm{pp}(u(x)) = 0$.

Gauss's Lemma and Lemma 5 together imply

$$\begin{aligned} \mathrm{cont}(uv) &= a\,\mathrm{cont}(u)\,\mathrm{cont}(v), \\ \mathrm{pp}(u(x)v(x)) &= b\,\mathrm{pp}(u(x))\,\mathrm{pp}(v(x)), \end{aligned} \tag{1}$$

where a, b are inverse units in S ($ab = 1$) that depend on u, v. The only units in $\mathbb{Z}$ are ± 1 and for polynomials over $\mathbb{Z}$ it is conventional to define $\mathrm{pp}(u(x))$ so that its leading coefficient is positive and hence $\mathrm{cont}(u)$ is the gcd of

the coefficients of $u(x)$ with the sign of the leading coefficient. All nonzero elements of a field are units and for polynomials over a field it is conventional to define $\mathrm{pp}(u(x))$ so that its leading coefficient is monic and hence $\mathrm{cont}(u)$ is the leading coefficient of $u(x)$. With these definitions, relations (1) hold with $a = b = 1$ for all u, v.

However, since there are no prime elements in a field, no prime can divide the coefficients of *any* polynomial over a field, and hence every polynomial over a field is primitive. Therefore, one is free to choose some other rule for splitting a polynomial over a field into its content and primitive part. For example, the definition of content of a polynomial over the field of fractions of an integral domain can be used, as given in the previous set of notes, which has the advantage for some purposes that the primitive part is a polynomial over an integral domain rather than a field, and an integral domain that is not a field has non-trivial homomorphic images that can be used to facilitate some computations.

2.2 GCDs in terms of content and primitive part

The following proposition splits the polynomial gcd problem into a univariate polynomial problem and a coefficient domain problem.

Proposition 6

$$\begin{aligned} \mathrm{cont}(\gcd(u,v)) &= a\,\gcd(\mathrm{cont}(u), \mathrm{cont}(v)), \\ \mathrm{pp}(\gcd(u(x)v(x))) &= b\,\gcd(\mathrm{pp}(u(x)), \mathrm{pp}(v(x))), \end{aligned}$$

where a and b are units.

Proof If $g = \gcd(u, v)$ then $u = gu', v = gv'$ where $\gcd(u', v') = 1$. Then by (1),

$$\mathrm{cont}(u) = a\,\mathrm{cont}(g)\,\mathrm{cont}(u'), \quad \mathrm{cont}(v) = a\,\mathrm{cont}(g)\,\mathrm{cont}(v'),$$

and hence

$$\gcd(\mathrm{cont}(u), \mathrm{cont}(v)) = a\,\mathrm{cont}(g),$$

because $\gcd(u', v') = 1 \;\Rightarrow\; \gcd(\mathrm{cont}(u'), \mathrm{cont}(v')) = 1$. The argument for the primitive part is identical. □

Note that the content of the polynomial gcd on the left, $\gcd(u(x)v(x))$, is irrelevant because it is discarded, and the polynomial gcd computation is

reduced to one of finding a gcd of primitive polynomials. Proposition 6 provides a way of computing gcds of polynomials over UFDs that do not form Euclidean domains by performing the polynomial division over the field of fractions of the coefficient domain. To compute $\gcd(u, v)$, where $u, v \in S[x]$ and S is a UFD, compute $\mathrm{cont}(u), \mathrm{cont}(v)$ as the gcds of their coefficients and then compute $\gcd(\mathrm{cont}(u), \mathrm{cont}(v))$, which requires only gcd computations in S. Then compute $\gcd(\mathrm{pp}(u(x)), \mathrm{pp}(v(x)))$ over $Q(S)$, discard any common denominator, take the primitive part of the result over S and multiply it by the content, found separately.

For example, let us use this technique to compute $\gcd(6x^2y, 15xy^2 + 21x^3y^2)$ in $\mathbb{Z}[y][x]$. Over $\mathbb{Z}[y]$, $\mathrm{cont}(6x^2y) = 6y$, $\mathrm{cont}(15xy^2 + 21x^3y^2) = 3y^2$, and their gcd is $3y$. The primitive parts are respectively x^2 and $5x + 7x^3$, the gcd of which is x, and hence the overall gcd is $3yx$.

2.3 Multivariate GCD computation

If the coefficient domain is itself a ring of polynomials then the polynomial gcd algorithm is called recursively, until finally the coefficient domain is numerical. Suppose that we have a procedure Euclid0(u, v) that computes the (distinguished) gcd of two elements u, v of the ground ring. In practice, this will normally be either the integers, which form a Euclidean domain, or a field in which case the (distinguished) gcd is trivially 1. Suppose also that we have a procedure Euclid1(u, v, x) that computes a gcd of $u, v \in S[x]$ by using any convenient method, such as computing over the field of fractions of the coefficient domain, and returns a gcd in $S[x]$ with arbitrary content. Later we will consider various alternative ways that Euclid1 could operate. Then, in terms of these two Euclidean procedures that do the real gcd computation, a procedure to recursively compute the gcd of two polynomials in r variables can be written as follows.

input: $u, v \in R[x_1, \ldots, x_r]$, R a UFD.
procedure $\gcd(u, v, r)$;
if $r = 0$ then return Euclid0(u, v) else
begin
    $u_c := \mathrm{cont}(u, r)$; $u_p := u/u_c$;
    $v_c := \mathrm{cont}(v, r)$; $v_p := v/v_c$;
    return $\gcd(u_c, v_c, r-1) \times \mathrm{pp}(\text{Euclid1}(u_p, v_p, x_r), r)$
end;
output: $\gcd(u, v) \in R[x_1, \ldots, x_r]$

procedure $\mathrm{pp}(u, r)$; $u/\mathrm{cont}(u, r)$;

input: $u \in R[x_1, \ldots, x_{r-1}][x_r]$, R a UFD.
procedure $\mathrm{cont}(u, r)$;
begin
    $c := \mathrm{coeff}(u, x_r, 0)$; $i := 1$;
    while $i < \deg(u, x_r)$ and $c \neq 1$ do
    begin
        $c := \gcd(c, \mathrm{coeff}(u, x_r, i), r - 1)$;
        $i := i + 1$
    end;
    return c
end;
output: $\mathrm{cont}(u) \in R[x_1, \ldots, x_{r-1}]$

Because only the primitive part of the polynomial returned by Euclid1 is used, the contents of the polynomials supplied to it as arguments are irrelevant, but making the argument polynomials primitive as above reduces the complexity of the input. If Euclid1 is arranged always to return the primitive part of the gcd then procedure pp in the above algorithm is unnecessary.

Procedure content assumes the existence of an operator $\mathrm{coeff}(u, x, i)$ that returns the coefficient of x^i in the polynomial $u(x)$, and for efficiency the loop stops as soon as a content of 1 is detected. If u is a constant then this procedure returns $\mathrm{cont}(u, r) = u$, and hence $\mathrm{pp}(u, r) = 1$. Therefore, the procedures work correctly when one of u and v is a constant, although they could perhaps be made more efficient by including some special-case code.

The above discussion provides an approach to general gcd computation that works. However, it has the serious problems elaborated in Davenport *et al.* that it is highly recursive, the sizes of integers appearing in intermediate coefficients become very large, and the degrees of intermediate polynomials in the multivariate case also become very large. Nevertheless, it provides a basis for more sophisticated methods such as those using modular arithmetic.

3 Polynomial remainder sequences (PRSs)

The main purpose of this section is to consider alternatives to computing over the field of fractions of the coefficients, which requires many gcd computations, when applying Euclid's algorithm to polynomials.

Euclid's algorithm applied to two univariate polynomials $p_1(x), p_2(x)$, $\deg p_2(x) < \deg p_1(x)$, over a field generates a sequence of polynomials, each of which is the remainder of the previous two. This is called a *Euclidean sequence* or a *polynomial remainder sequence*, abbreviated to PRS. The general formulation is that the remainder sequence $\{p_i(x)\}, i \geq 3$ is generated by

$$p_i(x) = p_{i+1}(x)q_i(x) + p_{i+2}(x), \quad \deg p_{i+2}(x) < \deg p_{i+1}(x), \quad 1 \leq i \leq h-1,$$
$$p_{h+1}(x) = 0, \quad \gcd(p_1(x), p_2(x)) = p_h(x).$$

But this applies only to polynomials over a field, which form a Euclidean domain. For polynomials over a ring only pseudo-division rather than true polynomial division is generally possible, and pseudo-division leads to a *pseudo-remainder sequence*. This means that $p_i(x)$ on the left of the above division relation is multiplied by the constant needed to ensure that it is divisible by $p_{i+1}(x)$. However, it is also possible to rescale the result $p_{i+2}(x)$ before re-using it. This leads to a generalized PRS in which the division relation has the form

$$L_{i+1}{}^{d_i - d_{i+1}+1} p_i(x) = p_{i+1}(x)q_i(x) + \beta_i p_{i+2}(x),$$

where L_{i+1} represents the leading coefficient of $p_{i+1}(x)$ and $d_i = \deg p_i(x)$.

In general, the primitive part of a gcd determined by this generalized PRS will be correct, but its content will have been changed in a fairly arbitrary way by the rescalings. However, since we only need the primitive part this does not matter.

3.1 Alternative polynomial remainder sequences

The simplest polynomial pseudo-remainder sequence, which does not rescale $p_{i+2}(x)$ and so corresponds to taking $\beta_i = 1$, is usually called the *Euclidean PRS*. It has the disadvantage that the polynomial coefficients generally increase in length exponentially with respect to the degrees of the initial polynomials. One solution is to divide out the content of each polynomial and replace it by its primitive part, which corresponds (loosely) to taking $\beta_i = \mathrm{cont}(p_{i+2}(x))$, to give a *primitive PRS*. This gives the least possible coefficient growth, but has the disadvantage that it involves a lot of gcd computations to compute the contents.

The best compromise is obtained from an algorithm usually called the *sub-resultant* algorithm, which corresponds to a rescaling such that the coefficient growth is only linear, and no gcd computations are required. The

result is usually called a *sub-resultant PRS*. The method was discovered independently and published by Collins in 1967 and Brown in 1971, but it is related to work by Sylvester (on resultants) in 1853 which was generalized by Habicht in 1948. It is the best gcd algorithm based on Euclid's algorithm, although more advanced modular methods are better still. (It is discussed in great detail by Akritas, together with an alternative method of his own.)

The sub-resultant algorithm consists of using the following rescalings, i.e. dividing the factor β_i, defined as follows, out of $p_{i+2}(x)$ before re-using it. Let $\delta_i = \deg p_i(x) - \deg p_{i+1}(x)$ and $L_i = \mathrm{lc}\, p_i(x)$, and take

$$\beta_1 = (-1)^{\delta_1+1}, \quad \beta_i = (-1)^{\delta_i+1} L_i H_i^{\delta_i}, \ i \geq 2,$$

where

$$H_1 = 1, \quad H_i = L_i^{\delta_{i-1}} H_{i-1}^{1-\delta i-1}, \ i \geq 2.$$

A PRS is called *complete* if the degree of each element is one less than that of the previous element, so that $\forall i, \delta_i = 1$. This possibility is most important for Sturm sequences, to be discussed next week, but there is no way to determine *a priori* whether a PRS is going to be complete. However, if it is complete the above rescaling simplifies considerably to

$$\beta_1 = 1, \quad \beta_i = L_i^{\delta_i+1}, \ i \geq 2,$$

which corresponds to Sylvester's original algorithm; it can be used with an incomplete PRS but gives faster than optimal coefficient growth.

3.2 Examples of polynomial remainder sequences

The following examples were all computed by implementing in REDUCE 3.4 the algorithms described above, and applying them to the following pair of polynomials (as used in Davenport *et el.* §2.3.3, which I will abbreviate to DST):

$$\begin{aligned} p_1 &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5, \\ p_2 &= 3x^6 + 5x^4 - 4x^2 - 9x + 21. \end{aligned}$$

I will show the actual REDUCE output, which corresponds to the sequence p_3, p_4, p_5, p_6. The most elementary PRS is the genuine *remainder* sequence obtained by computing over the field of rational coefficients:

```
   5   4    1   2    1
- ---*x  + ---*x  - ---,
   9        9        3

   117   2          441
- -----*x  - 9*x + -----,
   25               25

 233150       102500
--------*x - --------,
 19773         6591

   1288744821
- ------------
   543589225
```

This output agrees with that in DST except for the linear element, where mine is 1/3 times that of DST – I do not know why.

All the subsequent sequences will be *pseudo-remainder* sequences; the next most sophisticated, called *Euclidean*, uses pseudo-division as defined last week with no additional rescaling, and gives

```
       4      2
 - 15*x  + 3*x  - 9

       2
 15795*x  + 30375*x - 59535

 1254542875143750*x - 1654608338437500

 125933387955007431009311419921875 00
```

This agrees with that in DST.

The smallest possible coefficients are obtained by making every polynomial primitive:

```
      4    2
 - 5*x  + x  - 3
```

```
    2
13*x  + 25*x - 49

4663*x - 6150

1
```

DST do not give this sequence, but it is obviously primitive, and each element is a (large) multiple of that in the Euclidean sequence.

Generally the algorithm with best overall performance, but the hardest to program, is the subresultant algorithm. It produces the following sequence:

```
    4      2
15*x  - 3*x  + 9

    2
65*x  + 125*x - 245

9326*x - 12300

260708
```

This agrees with DST, and is obviously a compromise between the Euclidean and primitive sequences. The theory behind the subresultant PRS algorithm is important because it is probably the main PRS used in practice, but it is quite complicated so I will defer consideration of it until next week.

4 Bézout's identity and its applications

Bézout's identity provides a systematic technique for computing inverse elements in quotient rings, and also leads to an elegant technique for computing partial fraction decompositions without needing to solve any equations. I will discuss these two applications after a brief look at the background theory.

Theorem 7 *Two integer a, b are relatively prime if and only if there exist integer s, t such that*

$$sa + tb = 1.$$

This relation is attributed to Étienne Bézout (1730–83), who was an algebraic geometer (although Mignotte claims that it should be attributed to Bachet de Méziriac).

Proof There exist $s, t \in \mathbb{Z}$ such that $\gcd(a, b) = sa + tb$, and hence necessity is proved. Sufficiency is proved by the fact that if the relation holds then any integer that simultaneously divides a and b must also divide 1, and hence a and b must be relatively prime. □

The above theorem has the following corollary:

Theorem 8 (Euclid-Gauss) *Let $a, b \in \mathbb{Z}$ be relatively prime. If $c \in \mathbb{Z}$ is divisible by both a and b then it is also divisible by ab.*

Proof $\exists s, t \in \mathbb{Z},\ sa + tb = 1$. Then $sac + tbc = c$, and since ab divides both ac and bc, it must divide c. □

Bézout's identity generalizes to any Euclidean domain, and in particular to polynomials over a field F. A slightly different point of view from that which we have taken so far ties in with ideals; recall that a Euclidean domain is a principal ideal domain (PID).

Theorem 9 *Let $f_1, \ldots, f_r \in F[x]$. Any generator g of the ideal $(f_1, \ldots, f_r)$ is a gcd of the f_i, and there exist polynomials $u_1, \ldots, u_r \in F[x]$ such that*

$$u_1 f_1 + \cdots + u_r f_r = g.$$

Moreover, g divides each f_i, and any polynomial p that divides each of the f_i also divides g. Any other gcd of the f_i has the form μg, where μ is a unit, i.e. any nonzero element, in F. Euclid's algorithm can be used to compute g, and extended to compute the u_i.

Hence:

Corollary 10 *If the polynomials $f_1, \ldots, f_r \in F[x]$ are relatively prime then there exist polynomials $u_1, \ldots, u_r \in F[x]$ such that Bézout's identity holds:*

$$u_1 f_1 + \cdots + u_r f_r = 1.$$

However, Bézout's identity is most often used with $r = 2$, as in the following two applications. The coefficients s, t in Bézout's identity are computed in practice by using the extended Euclidean gcd algorithm, although in simple examples they can often be guessed quite easily.

4.1 Computation of inverses in a quotient ring

Let D be a Euclidean domain. Then computation in the quotient ring $D/(m)$, where as usual (m) denotes a (principal) ideal, is the same as computation "mod m". The problem of computing inverses in $D/(m)$ is the same as finding solutions u of the congruence $au \equiv 1 \pmod{m}$ for given $a, m \in D$. A distinguished representative u of an equivalence class $[u]$ in $D/(m)$ is chosen so that $u = r_m(u)$.

4.1.1 Theory

The simplest Euclidean domain is $\mathbb{Z}$, in which the congruence $5u \equiv 1 \pmod{8}$ has the (distinguished) solution $u = 5$, whereas $6u \equiv 1 \pmod{8}$ has no solution. The reason is the following:

Theorem 11 *Let $a, m \in D$; then the congruence $au \equiv 1 \pmod{m}$ has a solution $u \in D$ if and only if $\gcd(a, m) = 1$, i.e. a and m are relatively prime.*

Proof Applying Bézout's identity, $\gcd(m, a) = 1 \Rightarrow \exists s, t \in D$ such that $sm + ta = 1$, and hence $u = t$ is a solution. Conversely, if the congruence has a solution then $au + mv = 1$ for some $v \in D$, and hence $\gcd(m, a) = 1$. □

Corollary 12 *In $D/(m)$, the element $[a] = a + (m)$ is invertible if and only if $\gcd(a, m) = 1$.*

From any particular solution we can compute all solutions:

Theorem 13 *If $v \in D$ is a particular solution of $au \equiv 1 \pmod{m}$ then u is a solution if and only if $u \equiv v \pmod{m}$.*

Proof Given that $av \equiv 1 \pmod{m}$ for some $v \in D$ then trivially $u \equiv v \pmod{m} \Rightarrow au \equiv 1 \pmod{m}$. Conversely, $au \equiv 1 \pmod{m} \Rightarrow au \equiv av \pmod{m}$, and hence $m \mid a(u - v)$. But $\gcd(a, m) = 1$, so $m \mid (u - v)$ and hence $u \equiv v \pmod{m}$. □

If $u \equiv v \pmod{m}$ then $u \bmod m = v \bmod m$, and hence $v \bmod m$ is the *unique* solution of $u \equiv v \pmod{m}$ in the range of the mod m function. This leads to

Corollary 14 *Let $m, a \in D$ with $\gcd(m, a) = 1$. Then $au \equiv 1 \pmod{m}$ has a unique solution in the range of the mod m function on D.*

This unique solution is denoted $a^{-1} \bmod m$.

Finally, the argument a can be reduced mod m before computing its inverse:

Corollary 15 $a^{-1} \bmod m = (a \bmod m)^{-1} \bmod m$.

Proof Clearly, $au \equiv 1 \pmod{m} \Rightarrow (a \bmod m)u \equiv 1 \pmod{m}$. Hence the unique solutions to the two congruences in the range of the mod m function must be the same. □

The case that m is a prime or irreducible element of D is especially important, because $\gcd(m, a) = 1 \iff m \nmid a$. This reflects the fact that $D/(m)$ is a field if m is prime, and in $D/(m)$, $[a] \neq [0] \iff m \nmid a$, in which case $[a]$ must be invertible.

4.1.2 Computation

The extended Euclidean gcd algorithm applied to $\{m, a\}$ returns $\{g, s, t\}$ such that

$$\gcd(m, a) \sim g = sm + ta.$$

If g is not a unit in D ($g \not\sim 1$) then, by Theorem 11, $a^{-1} \bmod m$ does not exist. Otherwise, $g \sim 1$ and hence

$$1 = g^{-1}sm + g^{-1}ta$$

and $a^{-1} \bmod m = g^{-1}t$.

Since the value of t is not required, the extended Euclidean algorithm can be streamlined slightly. The time to compute the inverse is essentially the time to perform Euclid's algorithm, which for "small" integers bounded in magnitude by N is $O(\lg N)$, and for polynomials of degree bounded by N is $O(N^2)$ assuming coefficient operations take constant $[O(1)]$ time.

Some particularly important applications are computations of inverses in the following fields:

- a^{-1} in the finite field $\mathbb{Z}_p$ for prime p, in which case it is guaranteed that $\gcd(p, a) = 1$ because $0 \leq a < p$;

- $a(x)^{-1}$ in the field $F[x]_{m(x)} \cong F[x]/(m(x))$ where $m(x)$ is irreducible over the field F and $a(x)$ is first reduced mod $m(x)$ so that $\deg a(x) < \deg m(x)$.

As an explicit example, $1/3 = -2 = 5$ in $\mathbb{Z}_7$ by Bézout's identity, a result which is clearly correct because $3 \cdot 5 = 15 = 1 \bmod 7$.

4.2 Partial fraction decomposition

This is a useful technique for simplifying a complicated fraction or rational expression n/d into a sum of simpler fractions, which can be useful when a fraction is to be integrated or summed, for example. If the fraction is improper, i.e. the degree of the numerator is not less than that of the denominator, then a Euclidean division can be performed leading to a polynomial quotient and a fractional remainder, so that the fraction can be assumed to be proper. The first step is to factorize the denominator into relatively prime factors – suppose that $d = pq$ where $\gcd(p, q) = 1$. Then the problem is to express n/d in the form

$$\frac{n}{d} = \frac{n}{pq} = \frac{a}{p} + \frac{b}{q}.$$

The standard technique is to construct a partial fraction decomposition with unknown coefficients, clear denominators and then solve for the coefficients. Bézout's identity provides an alternative approach.

We can use the extended Euclidean algorithm to compute P, Q such that (for relatively prime p, q) $1 = \gcd(p, q) = Pp + Qq$. This immediately gives the decomposition

$$\frac{n}{pq} = \frac{n(Pp + Qq)}{pq} = \frac{nQ}{p} + \frac{nP}{q}.$$

The partial fractions computed in this way will not necessarily be proper, but if not they can simply be divided out. If the initial fraction was proper then the improper parts of the partial fractions must cancel by subtraction, so that their numerators can simply be replaced by the remainders when they are divided by their denominators.

I assumed above that p, q were relatively prime but not necessarily that they were irreducible. If any denominator is reducible then the procedure can be repeated until a sum of partial fractions with irreducible denominators results, and this can be done systematically.

5 Exercises

The assessed questions in this set of exercises are the first three.

1. (** **Assessed** **)
   Find the squarefree decomposition of x^4+x^3 by implementing by hand both the Tobey-Horowitz algorithm and Musser's algorithm, showing all the steps. [The problem is intentionally trivial, so that it can be easily done by hand!]

2. (** **Assessed** **)
   Two polynomials $p, q \in \mathbb{Z}[x, y]$ are given by

   $$p = xy - x + y - 1, \quad q = x^2y + 2x^2 + 2xy + 4x + y + 2.$$

   Write them explicitly as elements of $(\mathbb{Z}[y])[x]$ and hence find their contents and primitive parts, regarding p, q as univariate polynomials in x. By quoting the appropriate result for expressing polynomial gcds in terms of content and primitive part, find $\gcd(p, q)$.

3. (** **Assessed** **)
   Use Bézout's identity to perform a partial fraction decomposition of $1/(x^3 - x)$, showing the details of your calculation.

4. Design a polynomial squarefree decomposition algorithm that operates over a coefficient ring that is not a field and returns the squarefree decomposition as a product of powers of primitive factors, and also returns the content without performing an explicit content computation.

5. Verify the polynomial remainder sequences quoted in the notes.

6. Implement the gcd algorithm for $\mathbb{Z}[x]$ given in the text using the various generalized PRSs described.

7. Find (by "inspection" or guessing) Bézout's identity relating 3 and 5, and hence compute $3^{-1} \in \mathbb{Z}_5$. Find similarly Bézout's identity relating $x+1$ and x^3-1 in $\mathbb{Z}[x]$, and hence compute $(x+1)^{-1}$ in $\mathbb{Z}[x]/(x^3-1)$. State whether this quotient ring is a field, and prove your assertion.